Chapter 2

Solution Methodologies

2.1 Mathematical Prerequisites

2.1.1 Crisp Set Theory

Crisp Set: By crisp one mean dichotomous, that is, yes or no type rather than more-or-less type. In conventional dual logic, for instance, a statement can be true or false – and nothing in between. In set theory, an element can either belongs to a set or not; and in optimization, a solution is either feasible or not. A classical set, X, is defined by crisp boundaries, i.e., there is no uncertainty in the prescription of the elements of the set. Normally it is defined as a well defined collection of elements or objects, $x \in X$, where X may be countable or uncountable.

Convex Set: A subset $S \subset \mathfrak{R}^n$ is said to be convex, if for any two points s_1, s_2 in S, the line segment joining the points s_1 and s_2 is also contained in S. Mathematically, a subset $S \subset \mathfrak{R}^n$ is convex, if and only if

$$\forall s_1, s_2 \in S \Rightarrow \lambda s_1 + (1-\lambda)s_2 \in S; \quad 0 \leq \lambda \leq 1$$

Convex Combination: Given a set of vectors $\{v_1, v_2, \cdots, v_n\}$, a linear combination $x = \lambda_1 v_1 + \lambda_2 v_2 + \cdots + \lambda_n v_n$ is called a convex combination of the given vectors, if $\lambda_1, \lambda_2, \cdots, \lambda_n \geq 0$ and $\sum_{i=1}^{n} \lambda_i = 1$.

Convex Function: The function $f : S \to \Re$ is said to be convex if for any $s_1, s_2 \in S$ and $0 \leq \lambda \leq 1$, implies that

$$f\{\lambda s_1 + (1-\lambda)s_2\} \leq \lambda f(s_1) + (1-\lambda) f(s_2)$$

The definition of convex functions can be modified for concave functions by replacing $' \leq'$ by $' \geq'$.

2.1.2 Fuzzy Set Theory

The membership value $\mu_A(x)$ of an element x with respect to the crisp set A is either 0 or 1, i.e., an object x either belongs to A ($\mu_A(x) = 1$) or does not belongs to A ($\mu_A(x) = 0$). But in fuzzy set theory, the membership value of an element can be any value in the interval $[0, 1]$ specified by a proper membership function, i.e., there is an ambiguity whether the object belongs to the set or not. This concept of fuzzy set was initialized by Prof. L.A. Zadeh [208] in 1965. The fuzzy set theory has been well developed and widely used in real life problems in SC/inventory control problems during last few decades [21, 66, 68, 88, 98, 99, 112, 113, 117].

2.1.2.1 Fuzzy Set

Fuzzy sets deal with objects that are 'matter of degree', with all possible grades of truth between yes or no, and the shades of grey between white and black. So a fuzzy set is a class of objects in which there is no sharp boundary between those objects that belong to the class and those that do not. Let X be a collection of objects and x be an element of X, then a fuzzy set $\widetilde{A}$ in X is a set of ordered pairs $\widetilde{A} = \{(x, \mu_{\widetilde{A}}(x))/x \in X\}$, where $\mu_{\widetilde{A}}(x)$ is called the membership function or grade of membership of x in $\widetilde{A}$ which maps X to the membership space M which is considered as the closed interval $[0, u]$, where $0 < u \leq 1$.

Note: When M consists of only two points 0 and 1, $\widetilde{A}$ becomes a non-fuzzy set (or crisp set) and $\mu_{\widetilde{A}}(x)$ reduces to the characteristic function of the non-fuzzy set (or crisp set).

- **Equality:** Two fuzzy sets $\widetilde{A}$ and $\widetilde{B}$ in X are said to be equal if and only if $\mu_{\widetilde{A}}(x) = \mu_{\widetilde{B}}(x), \forall x \in X$.

- **Containment:** A fuzzy set $\widetilde{A}$ in X is contained in or is a subset of another fuzzy set $\widetilde{B}$ in X, written as $\widetilde{A} \subset \widetilde{B}$ if and only if $\mu_{\widetilde{A}}(x) \leq \mu_{\widetilde{B}}(x), \forall x \in X$.

- **Support:** The support of a fuzzy set $\widetilde{A}$ is a crisp set, denoted by $S(\widetilde{A})$ and is defined as

$$S(\widetilde{A}) = \{x \,|\, \mu_{\widetilde{A}}(x) > 0\}$$

- **Height:** The height of a fuzzy set $\widetilde{A}$ in X is the maximum membership grade value of $\widetilde{A}$, denoted by $h(\widetilde{A})$ and is defined as

$$h(\widetilde{A}) = \sup_{x \in X} \mu_{\widetilde{A}}(x)$$

  where, X is the universal set.

- **Normal fuzzy set:** A fuzzy set $\widetilde{A}$ in X is called normal if its height is 1, i.e., if $h(\widetilde{A}) = \sup_{x \in X} \mu_{\widetilde{A}}(x) = 1$.

- **Core:** The core of a fuzzy set $\widetilde{A}$ is a set of all points with unit membership degree in $\widetilde{A}$, denoted by $Core(\widetilde{A})$ and is defined as

$$Core(\widetilde{A}) = \{x \in X \,|\, \mu_{\widetilde{A}}(x) = 1\}$$

- **Convexity:** A fuzzy set $\widetilde{A}$ in X is said to be convex if and only if for any $x_1, x_2 \in X$, the membership function of $\widetilde{A}$ satisfies the inequality

$$\mu_{\widetilde{A}}(\lambda x_1 + (1-\lambda)x_2) \geq min\{\mu_{\widetilde{A}}(x_1), \mu_{\widetilde{A}}(x_2)\}, \text{ for } 0 \leq \lambda \leq 1$$

2.1.2.2 Fuzzy Number

A fuzzy number represents a fuzzy set which takes all possible values of a real object where each value is associated with a membership value (grade). This membership value represents the possibility of being actual value of the object. For example, price of some products in the market, holding cost of the retailer etc. As a fuzzy number represents the numerical value of an real object, it must be a fuzzy set in real number $\mathfrak{R}$ and some of its elements must have membership value 1. So a fuzzy number is a fuzzy set $\widetilde{A}$ defined in the set of real numbers $\mathfrak{R}$ having the following properties [208]:

- $\widetilde{A}$ is a convex fuzzy set.
- $\widetilde{A}$ is a normal fuzzy set.

In 1980, Dubois and Prade [42] was proposed more generalised definition of a fuzzy number $\widetilde{A}$, where $\widetilde{A}$ consists of four real parameters a_1, a_2, a_3, a_4 and its membership function $\mu_{\widetilde{A}}(x)$ takes the following form (cf. Figure 2.1):

$$\mu_{\widetilde{A}}(x) = \begin{cases} 0 & \text{if } x \le a_1 \\ L(x) & \text{if } a_1 \le x \le a_2 \\ 1 & \text{if } a_2 \le x \le a_3 \\ R(x) & \text{if } a_3 \le x \le a_4 \\ 0 & \text{if } x \ge a_4 \end{cases}$$

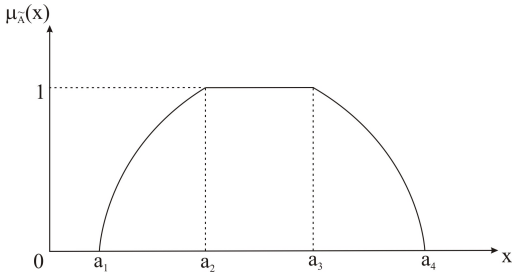

FIGURE 2.1: Graphical representation of a generalized fuzzy number

where, $L(x)$ and $R(x)$ are continuous function in their domain of definition, $L(x)$ is monotonic increasing and $R(x)$ is monotonic decreasing.

The fuzzy number $\widetilde{A}$ is said to be discrete or continuous according as its membership function $\mu_{\widetilde{A}}$ is discrete or continuous. Triangular Fuzzy Number (TFN) is a special class of continuous fuzzy number widely used in the modelling of different branches of science and engineering.

Triangular Fuzzy Number (TFN): A TFN $\widetilde{A}$ is specified by the triplet (a_1, a_2, a_3) and is defined by its continuous membership function $\mu_{\widetilde{A}}(x) : X \to [0, 1]$ as follows (cf. Figure 2.2):

$$\mu_{\widetilde{A}}(x) = \begin{cases} \dfrac{x - a_1}{a_2 - a_1} & \text{if } a_1 \le x \le a_2 \\ \dfrac{a_3 - x}{a_3 - a_2} & \text{if } a_2 \le x \le a_3 \\ 0 & \text{otherwise} \end{cases}$$

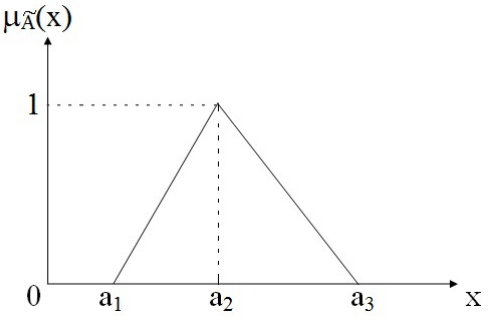

FIGURE 2.2: Membership function of TFN

Fuzzy Extension Principle [209]: Consider a real valued function $f(x, y)$ of two real variables x, y and $\widetilde{A}$, $\widetilde{B}$ be two fuzzy numbers. Then $f(\widetilde{A}, \widetilde{B})$ is another fuzzy number $\widetilde{C}$ having membership function $\mu_{\widetilde{C}}$ represented by

$$\mu_{\widetilde{C}}(z) = \sup_{x,y \in \Re} \{min(\mu_{\widetilde{A}}(x), \mu_{\widetilde{B}}(y)), z = f(x, y)\} \tag{2.2}$$

2.1.2.3 α-cut of a Fuzzy Number

α-cut of a fuzzy number $\widetilde{A}$ in X is denoted by $A[\alpha]$ and is defined as the following crisp set (cf. Figure 2.3):

$$A[\alpha] = \{x : \mu_{\widetilde{A}}(x) \geq \alpha, x \in X\}, \text{ where } \alpha \in [0, 1]$$

$A[\alpha]$ is a non-empty bounded closed interval contained in X and it can be denoted by $A[\alpha] = [A_L(\alpha), A_R(\alpha)]$. $A_L(\alpha)$ and $A_R(\alpha)$ are the lower and upper bounds of the closed interval respectively, which are the functions of α.

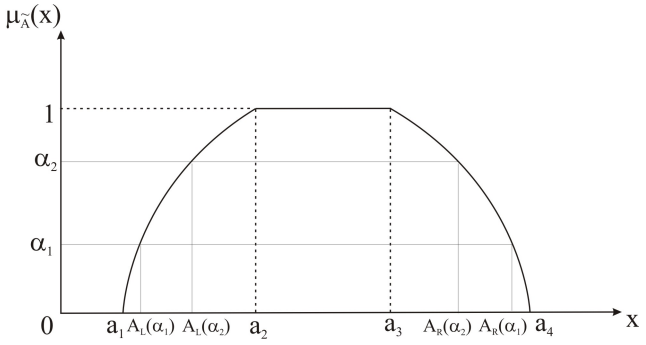

FIGURE 2.3: α-cut of generalized fuzzy number $\widetilde{A} = (a_1, a_2, a_3, a_4)$

Figure 2.3 represents a fuzzy number $\widetilde{A}$ with α-cuts $A[\alpha_1] = [A_L(\alpha_1), A_R(\alpha_1)]$, $A[\alpha_2] = [A_L(\alpha_2), A_R(\alpha_2)]$. It shows that if $\alpha_2 \geq \alpha_1$, then $A_L(\alpha_2) \geq A_L(\alpha_1)$ and $A_R(\alpha_2) \leq A_R(\alpha_1)$.

In particular, to find the α-cut for a TFN $\widetilde{A} = (a_1, a_2, a_3)$, it can be use the left and right reference function (cf. Figure 2.2) of $\widetilde{A}$; i.e., $\alpha = \dfrac{x - a_1}{a_2 - a_1}$ and $\alpha = \dfrac{a_3 - x}{a_3 - a_2}$, which gives the α-cut of $\widetilde{A}$ as $A[\alpha] = [a_1 + \alpha(a_2 - a_1), a_3 - \alpha(a_3 - a_2)]$.

α-cut of a function: Let $\widetilde{F}(X)$ be the space of all compact and convex fuzzy sets on X. If $f : \Re^n \to \Re$ is a continuous function, then $\widetilde{f} : \widetilde{F}(\Re^n) \to \widetilde{F}(\Re)$ is well

defined function and its α-cut $\widetilde{f}(u)[\alpha]$ is given by [159]

$$\widetilde{f}(u)[\alpha] = f(u[\alpha]),\ \forall \alpha \in [0,1], \forall \widetilde{u} \in \widetilde{F}(\Re^n) \tag{2.3}$$

where $f(A) = \{f(a)/a \in A\}$.

2.1.2.4 Defuzzification Methods

In the literature of fuzzy mathematics, several approaches are available to convert a fuzzy number into its equivalent crisp number [30, 31, 64, 205]. Each method has some merits and demerits over the others. In this thesis, used defuzzification method is discussed below.

Graded Mean Integration Value (GMIV) of Fuzzy Number: Chen and Hsieh [30, 31] introduced GMIV method based on the integral value of graded mean α-level of generalized fuzzy number. Suppose $\widetilde{A} = (a_1, a_2, a_3, a_4)$ is a generalized fuzzy number as shown in Figure 2.1. Then according to Chen et al. [30], Graded Mean Integration Value (GMIV) of $\widetilde{A}$ is denoted by $G(\widetilde{A})$ and is defined as

$$\begin{aligned} G(\widetilde{A}) &= \int_0^1 x\Big\{(1-\omega)L^{-1}(x) + \omega R^{-1}(x)\Big\}dx \Big/ \int_0^1 x dx \\ &= 2\int_0^1 x\Big\{(1-\omega)L^{-1}(x) + \omega R^{-1}(x)\Big\}dx \end{aligned}$$

where, $\omega \in [0,1]$ is a preassigned parameter called '*degree of optimism*'. Using this rule, GMIV of a TFN $\widetilde{B} = (b_1, b_2, b_3)$ is obtained as

$$G(\widetilde{B}) = \frac{1}{3}[(1-\omega)b_1 + 2b_2 + \omega b_3]$$

.

2.1.2.5 Different Measures in Fuzzy Environment

Let $\widetilde{A}$ and $\widetilde{B}$ be two fuzzy numbers with respective membership functions $\mu_{\widetilde{A}}(x)$ and $\mu_{\widetilde{B}}(x)$. Then for any comparison operator $\star \in \{>, <, =, \geq, \leq\}$, $\widetilde{A} \star \widetilde{B}$ represents a fuzzy event. In fact, for any fuzzy number $\widetilde{C}$ having membership function $\mu_{\widetilde{C}}(x)$, the degree of membership of an element $x \in X$ in the fuzzy set $\widetilde{C}$ can be

measured following any one of the three semantics of fuzzy numbers, namely degree of similarity proposed by Bellman et al. [8], degree of preference introduced by Bellman and Zadeh [9], degree of uncertainty proposed by Zadeh [209]. Following degree of uncertainty as the semantics of fuzzy numbers, different measures of the fuzzy events are proposed by several researchers in the last few decades [48, 105, 113, 117]. Each of these approaches has some merits and demerits over the others. Some of these approaches are presented here which are followed in this research work.

Possibility measure of a fuzzy event: The possibility measure of the fuzzy event $\widetilde{A} \star \widetilde{B}$ involving two fuzzy numbers $\widetilde{A}$ and $\widetilde{B}$, is denoted by $Pos(\widetilde{A} \star \widetilde{B})$ and is defined as

$$Pos(\widetilde{A} \star \widetilde{B}) = \sup\{\min(\mu_{\widetilde{A}}(x), \mu_{\widetilde{B}}(y)), x, y \in \Re, x \star y\} \tag{2.4}$$

where, $\star \in \{>, <, =, \geq, \leq\}$. This approach is normally followed by optimistic DMs only.

Necessity measure of a fuzzy event: The necessity measure of the fuzzy event $\widetilde{A} \star \widetilde{B}$ is denoted by $Nes(\widetilde{A} \star \widetilde{B})$ and is defined as

$$Nes(\widetilde{A} \star \widetilde{B}) = 1 - Pos(\overline{\widetilde{A} \star \widetilde{B}}) \tag{2.5}$$

where, complement of $\widetilde{A} \star \widetilde{B}$ is denoted by $\overline{\widetilde{A} \star \widetilde{B}}$. This approach is normally followed by pessimistic DMs only.

Credibility measure of a fuzzy event: The credibility measure of the fuzzy event $\widetilde{A} \star \widetilde{B}$ is denoted by $Cr(\widetilde{A} \star \widetilde{B})$ and is defined as

$$Cr(\widetilde{A} \star \widetilde{B}) = \frac{1}{2}\Big[Pos(\widetilde{A} \star \widetilde{B}) + Nes(\widetilde{A} \star \widetilde{B})\Big] \tag{2.6}$$

This measure is followed by most of the realistic DMs.

Lemma 2.1. If $\widetilde{A} = (a_1, a_2, a_3)$ and $\widetilde{B} = (b_1, b_2, b_3)$ be TFNs, then (cf. Liu [105])

$$\text{Cr } (\widetilde{A} \geq \widetilde{B}) = \begin{cases} 1 & \text{if } \ a_1 \geq b_3 \\ \dfrac{b_3 + 2(a_2 - b_2) - a_1}{2(b_3 - b_2 + a_2 - a_1)} & \text{if } \ a_2 \geq b_2, a_1 \leq b_3 \\ \dfrac{a_3 - b_1}{2(a_3 - a_2 + b_2 - b_1)} & \text{if } \ a_2 \leq b_2, a_3 \geq b_1 \\ 0 & \text{otherwise} \end{cases} \tag{2.7}$$

Proof.

$$\begin{aligned} Pos(\widetilde{A} \geq \widetilde{B}) &= \sup\{\min(\mu_{\widetilde{A}}(x), \mu_{\widetilde{B}}(y)), x, y \in \Re, x \geq y\} \\ Nes(\widetilde{A} \geq \widetilde{B}) &= 1 - Pos(\overline{\widetilde{A} \geq \widetilde{B}}) \end{aligned}$$

$$Cr(\widetilde{A} \geq \widetilde{B}) = \frac{1}{2}\Big[Pos(\widetilde{A} \geq \widetilde{B}) + Nes(\widetilde{A} \geq \widetilde{B})\Big]$$

(i) *for* $a_1 \geq b_3$ *(cf. Figure 2.4):*

$Pos(\widetilde{A} \geq \widetilde{B}) = 1$
$Nes(\widetilde{A} \geq \widetilde{B}) = 1 - Pos(\widetilde{A} < \widetilde{B})$
$= 1 - 0 = 1$
$Hence, Cr(\widetilde{A} \geq \widetilde{B}) = \frac{1}{2}(1+1) = 1$

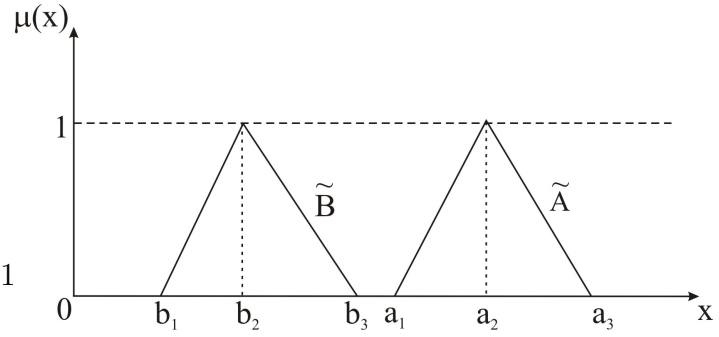

FIGURE 2.4: Membership functions

(ii) *for* $a_2 \geq b_2, a_1 \leq b_3$ *(cf. Figure 2.5):*

$Pos(\widetilde{A} \geq \widetilde{B}) = 1$
$Nes(\widetilde{A} \geq \widetilde{B}) = 1 - Pos(\widetilde{A} < \widetilde{B})$
$= 1 - \dfrac{b_3 - a_1}{a_2 - a_1 + b_3 - b_2}$
$= \dfrac{a_2 - b_2}{a_2 - a_1 + b_3 - b_2}$
$Hence, Cr(\widetilde{A} \geq \widetilde{B})$
$= \frac{1}{2}\Big[1 + \dfrac{a_2 - b_2}{a_2 - a_1 + b_3 - b_2}\Big]$
$= \dfrac{b_3 + 2(a_2 - b_2) - a_1}{2(b_3 - b_2 + a_2 - a_1)}.$

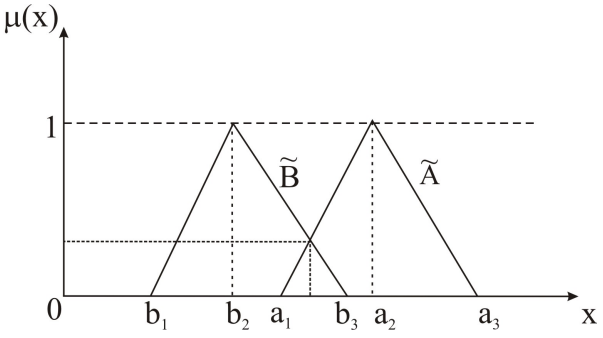

FIGURE 2.5: Membership functions

(iii) *for* $a_2 \leq b_2, a_3 \geq b_1$ *(cf. Figure 2.6):*

(iv) *In all other cases (cf. Figure 2.7):*

□

$$Pos(\widetilde{A} \geq \widetilde{B}) = \frac{a_3 - b_1}{b_2 - b_1 + a_3 - a_2}$$
$$Nes(\widetilde{A} \geq \widetilde{B}) = 1 - Pos(\widetilde{A} < \widetilde{B})$$
$$= 1 - 1 = 0$$
$$Hence, Cr(\widetilde{A} \geq \widetilde{B})$$
$$= \frac{a_3 - b_1}{2(a_3 - a_2 + b_2 - b_1)}.$$

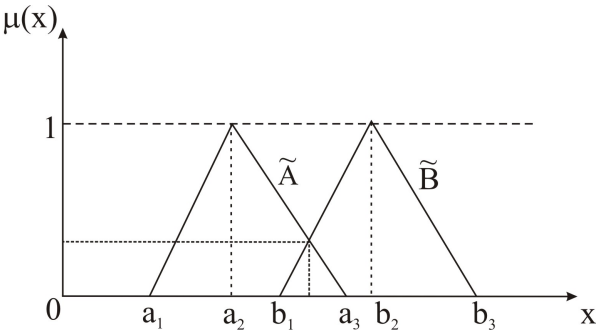

FIGURE 2.6: Membership functions

$$Pos(\widetilde{A} \geq \widetilde{B}) = 0$$
$$Nes(\widetilde{A} \geq \widetilde{B}) = 1 - Pos(\widetilde{A} < \widetilde{B})$$
$$= 1 - 1 = 0$$
$$Hence, Cr(\widetilde{A} \geq \widetilde{B}) = 0.$$

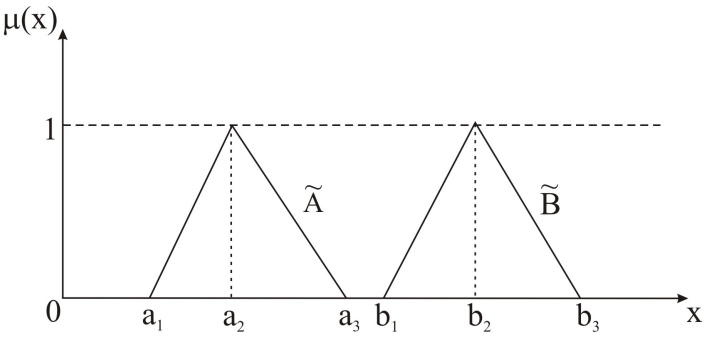

FIGURE 2.7: Membership functions

Lemma 2.2. If $\widetilde{A} = (a_1, a_2, a_3)$ be a TFN with non-negative components and b be any crisp number, then

$$\text{Cr } (\widetilde{A} \geq b) = \begin{cases} 1 & \text{if } b \leq a_1 \\ \dfrac{2a_2 - a_1 - b}{2(a_2 - a_1)} & \text{if } a_1 \leq b \leq a_2 \\ \dfrac{a_3 - b}{2(a_3 - a_2)} & \text{if } a_2 \leq b \leq a_3 \\ 0 & \text{otherwise} \end{cases} \tag{2.8}$$

Proof. This lemma can easily be proved, by putting $b_1 = b_2 = b_3 = b$ in Lemma 2.1. □

Lemma 2.3. If $\widetilde{A} = (a_1, a_2, a_3)$ be a TFN with non-negative components and b be any crisp number, then

$$\text{Cr } (\widetilde{A} \leq b) = \begin{cases} 1 & \text{if } b \geq a_3 \\ \dfrac{a_3 - 2a_2 + b}{2(a_3 - a_2)} & \text{if } a_2 \leq b \leq a_3 \\ \dfrac{b - a_1}{2(a_2 - a_1)} & \text{if } a_1 \leq b \leq a_2 \\ 0 & \text{otherwise} \end{cases} \tag{2.9}$$

Proof. Similar proof as Lemma 2.2. □

Lemma 2.4. The fuzzy constraint $\widetilde{A} > \widetilde{B}$ is necessarily and sufficiently true, if the credibility of the constraint, i.e., $Cr(\widetilde{A} > \widetilde{B}) > 0.5$, since $Cr(\widetilde{A} > \widetilde{B}) + Cr(\overline{\widetilde{A} > \widetilde{B}}) = 1$.

Proof. It follows from Lemma 2.1. □

Lemma 2.5. If $\widetilde{A} = (a_1, a_2, a_3)$ and $\widetilde{B} = (b_1, b_2, b_3)$ are TFNs, then $Cr(\widetilde{A} > \widetilde{B}) > 0.5$, iff $a_2 > b_2$.

Proof. It follows from Lemma 2.1. □

Fuzzy Expectation [106]: Let $\widetilde{X}$ be any normalized fuzzy variable. The expected value of the fuzzy variable $\widetilde{X}$ is denoted by $E[\widetilde{X}]$ and is defined by

$$E[\widetilde{X}] = \int_0^\infty Cr(\widetilde{X} \geq r)\, dr - \int_{-\infty}^0 Cr(\widetilde{X} \leq r)\, dr \tag{2.10}$$

provided that at least one of the two integral is finite.
Lemma 2.6. If $\tilde{\xi} = (a_1, a_2, a_3)$ is a TFN, then the expected value of $\tilde{\xi}$ is

$$E[\tilde{\xi}] = \frac{1}{4}(a_1 + 2a_2 + a_3) \tag{2.11}$$

Proof. Let t be a crisp number. Since $\tilde{\xi} = (a_1, a_2, a_3)$ is a TFN, then it satisfies the Eqns. (2.8) and (2.9). So, the expected value of $\tilde{\xi}$ can be calculated using (2.10) as follows:

$$\begin{aligned} E[\tilde{\xi}] &= \int_0^\infty Cr(\tilde{\xi} \geq t)\, dt - \int_{-\infty}^0 Cr(\tilde{\xi} \leq t)\, dt \\ &= \int_0^{a_1} 1\, dt + \int_{a_1}^{a_2} \frac{2a_2 - a_1 - t}{2(a_2 - a_1)}\, dt + \int_{a_2}^{a_3} \frac{a_3 - t}{2(a_3 - a_2)}\, dt \\ &= \frac{1}{4}(a_1 + 2a_2 + a_3) \end{aligned}$$

□

2.1.2.6 Fuzzy Differential Equation (FDE)

There are several approaches in the literature [19] to define fuzzy derivative. The concept of fuzzy differentiation was introduced by Dubois and Prade [45].

Motivated by their study [43–45], Seikkala [169] defines fuzzy differentiation and fuzzy integration using inclusion property of α-cuts of fuzzy numbers.

Definition: According to Seikkala [169], if $\tilde{Y}(t)$ be a fuzzy number for each $t \in I(\subseteq \Re)$, having α-cut, $\tilde{Y}(t)[\alpha] = [Y_L(t,\alpha), Y_R(t,\alpha)]$ then $\frac{d\tilde{Y}(t)}{dt}$ exists and $\frac{d\tilde{Y}(t)}{dt}[\alpha] = [\frac{dY_L(t,\alpha)}{dt}, \frac{dY_R(t,\alpha)}{dt}]$ provided $[\frac{dY_L(t,\alpha)}{dt}, \frac{dY_R(t,\alpha)}{dt}]$ are α-cuts of a fuzzy number for each $t \in I$, i.e., if the following conditions hold:

$$\begin{cases} \dfrac{dY_L(t,\alpha)}{dt} \text{ and } \dfrac{dY_R(t,\alpha)}{dt} \text{ are continuous on } I \times [0,1]. \\ \dfrac{dY_L(t,\alpha)}{dt} \text{ is an increasing function of } \alpha \text{ for each } t \in I. \\ \dfrac{dY_R(t,\alpha)}{dt} \text{ is a decreasing function of } \alpha \text{ for each } t \in I. \\ \dfrac{dY_L(t,1)}{dt} \leq \dfrac{dY_R(t,1)}{dt}, \forall\, t \in I. \end{cases} \tag{2.12}$$

Accordingly, they define fuzzy integral $\int_a^b \widetilde{Y}(t)dt$ for all $a, b \in I$ having α-cut

$$\left(\int_a^b \widetilde{Y}(t)dt\right)[\alpha] = \left[\int_a^b Y_L(\alpha,t)dt, \int_a^b Y_R(\alpha,t)dt\right]$$

provided that the integrals on the right exist. This definition of fuzzy integration agrees with the definition of Dubois and Prade [43] and Wu [194].

Fuzzy Differential Equation-1 (FDE-1) [19]: Consider the first order ordinary differential equation

$$\frac{dY}{dt} = f(t, Y, k), \quad Y(0) = C \tag{2.13}$$

where $k = (k_1, k_2, ...k_n)$ is a vector of constants and t is in some interval I (closed and bounded) which contains zero. Let the Eqn. (2.13) has a unique solution

$$Y = g(t, k, C), \text{ for } t \in I,\ k \in K \subset \Re^n,\ C \in \Re \tag{2.14}$$

When $\widetilde{k} = (\widetilde{k}_1, \widetilde{k}_2, ...\widetilde{k}_n)$ is a vector of fuzzy numbers and $\widetilde{C}$ be another fuzzy number, then the Eqn. (2.13) reduces to the following FDE

$$\frac{d\widetilde{Y}}{dt} = f(t, \widetilde{Y}, \widetilde{k}), \quad \widetilde{Y}(0) = \widetilde{C} \tag{2.15}$$

assuming that derivative [19, 169] of the unknown fuzzy function $\widetilde{Y}(t)$ exists according to the above definition [188]. According to Buckley and Feuring [19],

$$\widetilde{Y}(t) = g(t, \widetilde{k}, \widetilde{C}) \tag{2.16}$$

is solution of (2.15), if its α-cut $\widetilde{Y}(t)[\alpha] = [Y_L(t,\alpha), Y_R(t,\alpha)]$ satisfies the following conditions (2.17) along with the conditions given by the Eqn. (2.12).

$$\begin{cases} \dfrac{dY_L(t,\alpha)}{dt} = f_L(t,\alpha), \ \dfrac{dY_R(t,\alpha)}{dt} = f_R(t,\alpha), \ \forall \alpha \in [0,1]. \\ \dfrac{dY_L(0,\alpha)}{dt} = C_L(\alpha), \ \dfrac{dY_R(0,\alpha)}{dt} = C_R(\alpha), \ \forall \alpha \in [0,1]. \end{cases} \tag{2.17}$$

where, $\widetilde{f}(t)[\alpha] = [f_L(t,\alpha), f_R(t,\alpha)]$, $\widetilde{C}[\alpha] = [C_L(\alpha), C_R(\alpha)]$ and membership function of $\widetilde{Y}(t)$ is obtained using fuzzy extension principle (2.2). To justify the validity of this solution one can see [19].

Fuzzy Differential Equation-2 (FDE-2) [26]: Consider the fuzzy initial value problem

$$\tilde{X}'(t) = \tilde{f}(t, \tilde{X}(t)), \ \tilde{X}(0) = \tilde{X}_0 \tag{2.18}$$

where $f : [0,T] \times F(U) \to F(\mathfrak{R}^n)$ is obtained by Zadeh's extension principle (2.2) from a continuous function $g : [0,T] \times U \to \mathfrak{R}^n$, where $U \subset \mathfrak{R}^n$. As g is continuous so f is continuous [159] and by (2.3), we have

$$[f(t,X)](\alpha) = g(t, X[\alpha])$$

where, $g(t,A) = \{g(t,a)/a \in A\}$.

Consider the deterministic differential equation (DDE) associated with (2.18)

$$x'(t) = g(t, x(t)), \ x(0) = x_0 \tag{2.19}$$

where $x'(t)$ is the derivative (crisp) of a function $x : [0,T] \to \mathfrak{R}^n$. Then according to Chalco-Cano and Román-Flores [26], a fuzzy solution for (2.15) can be derived from (2.19) as follows

- Solve the DDE (2.19) and let $x(t, x_0)$ be its solution.

- Use Zadeh's [209] extension principle (2.2), to $x(t, x_0)$ in relation to the parameter x_0 and obtain the extension $\tilde{X}(t) = \tilde{x}(t, \tilde{X}_0)$, for each fixed t, which is a fuzzy solution of problem (2.18), provided conditions of following theorem (Theorem-1) hold.

Theorem-1: Let U be an open set in $\Re^n$ and $X_0[\alpha] \subset U$. Suppose that g is continuous and that for each $c \in U$, there exists a unique solution $x(., c)$ of the deterministic problem (2.19) and that $x(t, .)$ is continuous on U for each $t \in [0, T]$ fixed. Then, there exists a unique fuzzy solution $\widetilde{X}(t) = \tilde{x}(t, X_0)$ of the FDE (2.18).

2.1.2.7 Fuzzy Riemann Integration (FRI)

Fuzzy Integral: The study on fuzzy integral was started before three decades. Sims and Wang [174] gave a good review of this subject. Dubois and Prade [43] defined integral of fuzzy mapping $\widetilde{f}(x)$ over a crisp interval $I = [a, b]$ and proved that under certain condition $(\int_I \widetilde{f})[\alpha] = \int_I \widetilde{f}[\alpha]$. In a subsequent paper, Dubois and Prade [44] define integration of a real mapping $f(x)$ between fuzzy bounds $\widetilde{D} = [\widetilde{a}, \widetilde{b}]$. If $I = [I_L, I_R]$, where I_L is the infimum of the support of $\widetilde{a}$ and I_R is the supremum of the support of $\widetilde{b}$, then according to their definition

$$\forall z \in \Re,\ \mu_{\int_{\widetilde{D}}}(z) = \sup_{x,y \in I} min(\mu_{\widetilde{a}}(x), \mu_{\widetilde{b}}(y)), \text{ under the constraint } z = \int_x^y f(t)dt$$

But their definition does not include integration of fuzzy mapping over a fuzzy domain. In a subsequent paper, Wu [194] defined integration of fuzzy mapping over crisp and fuzzy intervals and accordingly two types of FRIs have been defined by him.

Fuzzy Riemann Integral of type-I [194]: Let $\tilde{f}(x)$ be a closed and bounded fuzzy valued function on $[a, b]$ and $[f_L(\alpha, x), f_R(\alpha, x)]$ be α-cut of $\tilde{f}(x)\ \forall x \in [a, b]$. If $f_L(\alpha, x)$ and $f_R(\alpha, x)$ are Riemann integrable on $[a, b]$, $\forall \alpha$, then the fuzzy Riemann integral $\int_a^b \tilde{f}(x)dx$ is a closed fuzzy number and its α-level set is given by

$$\left(\int_a^b \tilde{f}(x)dx\right)[\alpha] = \left[\int_a^b f_L(\alpha, x)dx, \int_a^b f_R(\alpha, x)dx\right]$$

Fuzzy Riemann Integral of type-II [194]: Let $\tilde{f}(\tilde{x})$ be a bounded and closed fuzzy valued function defined on the closed fuzzy interval $[\tilde{a}, \tilde{b}]$ and $\tilde{f}(x)$ be induced by $\tilde{f}(\tilde{x})$. $[f_L(\alpha, x), f_R(\alpha, x)]$ be α-cut of $\tilde{f}(x)$ and $\tilde{f}(x)$ is either nonnegative or nonpositive.

Case-1: If $\tilde{f}(x)$ is nonnegative and $f_L(\alpha, x)$ and $f_R(\alpha, x)$ are Riemann integrable on $[a_R(\alpha), b_L(\alpha)]$ and $[a_L(\alpha), b_R(\alpha)]$ respectively $\forall\alpha$, then the fuzzy Riemann integral $\int_{\tilde{a}}^{\tilde{b}} \tilde{f}(\tilde{x})d\tilde{x}$ is a closed fuzzy number and its α-level set is given by

$$\left(\int_{\tilde{a}}^{\tilde{b}} \tilde{f}(\tilde{x})d\tilde{x}\right)[\alpha] = \begin{cases} \left[\int_{a_R(\alpha)}^{b_L(\alpha)} f_L(\alpha, x)dx, \int_{a_L(\alpha)}^{b_R(\alpha)} f_R(\alpha, x)dx\right] & \text{if } b_L(\alpha) > a_R(\alpha) \\ \left[0, \int_{a_L(\alpha)}^{b_R(\alpha)} f_R(\alpha, x)dx\right] & \text{if } b_L(\alpha) \leq a_R(\alpha) \end{cases}$$

Case-2: If $\tilde{f}(x)$ is non positive and $f_L(\alpha, x)$ and $f_R(\alpha, x)$ are Riemann integrable on $[a_R(\alpha), b_L(\alpha)]$ and $[a_L(\alpha), b_R(\alpha)]$ respectively $\forall\alpha$, then the fuzzy Riemann integral $\int_{\tilde{a}}^{\tilde{b}} \tilde{f}(\tilde{x})d\tilde{x}$ is a closed fuzzy number and its α-level set is given by

$$\left(\int_{\tilde{a}}^{\tilde{b}} \tilde{f}(\tilde{x})d\tilde{x}\right)[\alpha] = \begin{cases} \left[\int_{a_L(\alpha)}^{b_R(\alpha)} f_L(\alpha, x)dx, \int_{a_R(\alpha)}^{b_L(\alpha)} f_R(\alpha, x)dx\right] & \text{if } b_L(\alpha) > a_R(\alpha) \\ \left[\int_{a_L(\alpha)}^{b_R(\alpha)} f_L(\alpha, x)dx, 0\right] & \text{if } b_L(\alpha) \leq a_R(\alpha) \end{cases}$$

2.1.3 Rough Set Theory

Rough set theory is also an excellent mathematical tool (like fuzzy set theory) to define an object in an imprecise environment. Rough set theory deals with vague description of objects, i.e., there are also some objects which have no such sufficient information to characterize the objects. The concept of rough set theory is introduced by Pawlak [144] in 1982. According to Pawlak [144], a rough set may be defined as a pair of two crisp sets, called the lower and upper approximations of the rough set, which are produced by an equivalence relation (reflexive, symmetric and transitive).

In 2000, the concept of rough set theory was extended by Slowinski and Vanderpooten [176] providing a binary similarity relation instead of equivalence relation.

According to Slowinski and Vanderpooten [176], a binary similarity relation has no symmetry and transitivity but has reflexivity.

Lower and Upper Approximation [176]: Let U be a universe, and X be a set representing a concept. Then its lower approximation is defined by

$$\underline{X} = \{x \in U \mid R^{-1}(x) \subset X\}; \tag{2.20}$$

while the upper approximation is defined by

$$\overline{X} = \bigcup_{x \in X} R(x) \tag{2.21}$$

where $R(x)$ is the similarity class of objects which are similar to x and $R^{-1}(x)$ is the class of objects to which x is similar.

Rough Set [144]: The collection of all sets having the same lower and upper approximations is called a rough set and is denoted by $(\underline{X}, \overline{X})$.

Rough Space [105]: Let Λ be a nonempty set, κ be a σ-algebra of subset of Λ and Δ be an element in κ and π nonnegative, real-valued, additive set function. Then $(\Lambda, \Delta, \kappa, \pi)$ is called a rough space.

Rough Variable [105]: A rough variable $\check{\xi}$ is a measurable function from the rough space $(\Lambda, \Delta, \kappa, \pi)$ to the set of real numbers, i.e., for every Borel set B of $\mathfrak{R}$, we have

$$\{\lambda \in \Lambda \mid \check{\xi}(\lambda) \in B\} \in \kappa$$

The lower and upper approximations of the rough variable are defined as $\underline{\check{\xi}} = \{\check{\xi}(\lambda) \mid \lambda \in \Delta\}$ and $\overline{\check{\xi}} = \{\check{\xi}(\lambda) \mid \lambda \in \Lambda\}$ respectively.

Trust Measure [105]: Let $(\Lambda, \Delta, \kappa, \pi)$ be a rough space. The trust measure of event A is denoted by $Tr\{A\}$ and defined by $Tr\{A\} = \frac{1}{2}(\underline{Tr}\{A\} + \overline{Tr}\{A\})$, where $\underline{Tr}\{A\}$ denotes the lower trust measure of event A, defined by $\underline{Tr}\{A\} = \frac{\pi\{A \cap \Delta\}}{\pi\{\Delta\}}$ and $\overline{Tr}\{A\}$ denotes the upper trust measure of event A, defined by $\overline{Tr}\{A\} = \frac{\pi\{A\}}{\pi\{\Lambda\}}$. When insufficient amount of information is given about the measurement of π for a real life problem, it may be viewed as the Lebesgue measure.

Let $\check{\xi} = ([a, b][c, d])$, $c \le a \le b \le d$ be a rough variable and Lebesgue measure is used for trust measure of an rough event associated with $\check{\xi} \ge t$. Then the trust

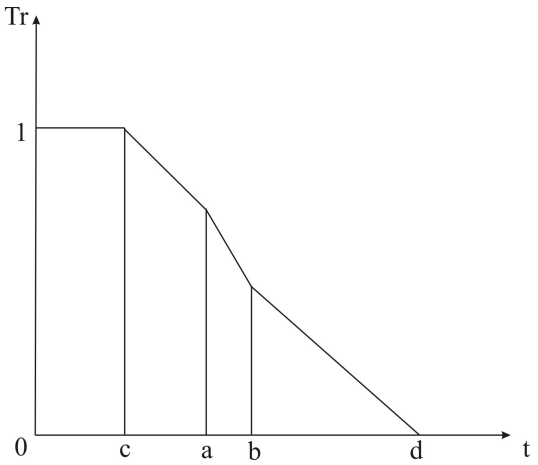

FIGURE 2.8: $\mathrm{Tr}\{\check{\xi} \geq t\}$ function

measure of the rough event $\check{\xi} \geq t$ is denoted by $Tr\{\check{\xi} \geq t\}$ and its function curve (cf. Figure 2.8) is presented below:

$$Tr\{\check{\xi} \geq t\} = \begin{cases} 0 & \text{for } d \leq t \\ \frac{d-t}{2(d-c)} & \text{for } b \leq t \leq d, \\ \frac{1}{2}\left(\frac{d-t}{d-c} + \frac{b-t}{b-a}\right) & \text{for } a \leq t \leq b, \\ \frac{1}{2}\left(\frac{d-t}{d-c} + 1\right) & \text{for } c \leq t \leq a, \\ 1 & \text{for } t \leq c \end{cases} \tag{2.22}$$

Also the trust measure of the rough event $\check{\xi} \leq t$ is denoted by $Tr\{\check{\xi} \leq t\}$ and its function curve (cf. Figure 2.9) is presented below:

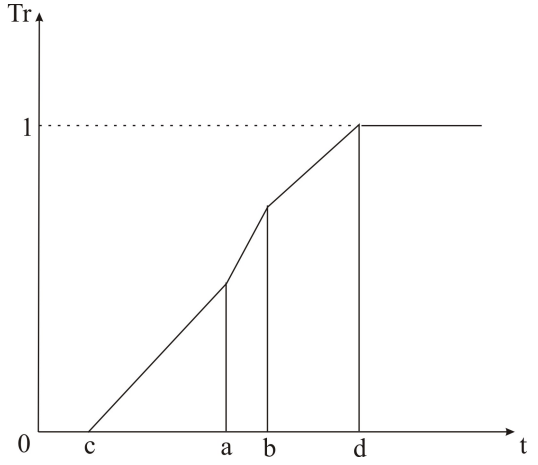

FIGURE 2.9: $\mathrm{Tr}\{\check{\xi} \leq t\}$ function

$$Tr\{\check{\xi} \le t\} = \begin{cases} 0 & \text{for } t \le c \\ \frac{t-c}{2(d-c)} & \text{for } c \le t \le a, \\ \frac{1}{2}\left(\frac{t-c}{d-c} + \frac{t-a}{b-a}\right) & \text{for } a \le t \le b, \\ \frac{1}{2}\left(\frac{t-c}{d-c} + 1\right) & \text{for } b \le t \le d, \\ 1 & \text{for } d \le t \end{cases} \tag{2.23}$$

Lemma 2.7. If $\check{A} = ([a_1, a_2][a_3, a_4])$ and $\check{B} = ([b_1, b_2][b_3, b_4])$ be rough variables, then according to Liu [105], Pramanik et al. [150]

$$Tr\{\check{A} \ge \check{B}\} = \begin{cases} 0 & \text{for } a_4 \le b_3 \\ \dfrac{a_4 - b_3}{2(a_4 - a_3 + b_4 - b_3)} & \text{for } a_2 \le b_1, b_3 \le a_4 \\ \dfrac{1}{2}\left[\dfrac{a_4 - b_3}{a_4 - a_3 + b_4 - b_3} + \dfrac{a_2 - b_1}{a_2 - a_1 + b_2 - b_1}\right] & \text{for } a_1 \le b_2, b_1 \le a_2 \\ \dfrac{1}{2}\left[\dfrac{a_4 - b_3}{a_4 - a_3 + b_4 - b_3} + 1\right] & \text{for } a_3 \le b_4, b_2 \le a_1 \\ 1 & \text{for } b_4 \le a_3 \end{cases} \tag{2.24}$$

Lemma 2.8. The rough constraint $\check{A} > \check{B}$ is necessarily and sufficiently true, if the trust measure of the constraint, i.e., $Tr(\check{A} > \check{B}) > 0.5$, since $Tr(\check{A} > \check{B}) + Tr(\overline{\check{A} > \check{B}}) = 1$.

Proof. It follows from Lemma 2.7. □

Lemma 2.9. For any two rough variables $\check{A} = ([a_1, a_2][a_3, a_4])$ and $\check{B} = ([b_1, b_2][b_3, b_4])$, if $a_1 > b_1$, $a_2 > b_2$, $a_3 > b_3$, $a_4 > b_4$ holds, then $Tr(\check{A} > \check{B}) > 0.5$.

Proof. It follows from Lemma 2.7. □

Rough Expectation [105]: let $\check{X}$ be a rough variable. The the expected value of the rough variable $\check{X}$ is denoted by $E[\check{X}]$ and is defined by

$$E[\check{X}] = \int_0^\infty Tr(\check{X} \ge r)\, dr - \int_{-\infty}^0 Tr(\check{X} \le r)\, dr \tag{2.25}$$

provided that at least one of the two integrals is finite.

Lemma 2.10. [105] Let $\check{\xi} = ([a, b][c, d])$ be a rough variable, where $c > 0$. Then expected value of $\check{\xi}$ is

$$E[\check{\xi}] = \frac{1}{4}(a + b + c + d) \tag{2.26}$$

Proof. Let t be a crisp number. Since $\check{\xi} = ([a,b][c,d])$ is a rough variable, then it satisfies the trust measure function curve (2.22) and (2.23). So, the expected value of $\check{\xi}$ can be calculated using (2.25) as follows:

$$\begin{aligned} E[\check{\xi}] &= \int_0^\infty Tr(\check{\xi} \geq t)\, dt - \int_{-\infty}^0 Tr(\check{\xi} \leq t)\, dt \\ &= \int_0^c 1\, dt + \int_c^a \frac{1}{2}\Big(\frac{d-t}{d-c} + 1\Big)\, dt + \int_a^b \frac{1}{2}\Big(\frac{d-t}{d-c} + \frac{b-t}{b-a}\Big)\, dt + \int_b^d \frac{d-t}{2(d-c)}\, dt \\ &= \frac{1}{4}(a+b+c+d) \end{aligned}$$

□

2.1.4 Interval and some useful properties

An interval I in $\Re$ is a subset of $\Re$ having two bounds I_L, I_R and defined as $I = \{x \in \Re | I_L \leq x \leq I_R\}$. I_L and I_R are termed as left and right bounds of I respectively and the interval is represented by $I = [I_L, I_R]$. It's mean and half-width (for simplicity, it is called width) are denoted by $m(I)$ and $w(I)$ respectively and are defined as $m(I) = (I_L + I_R)/2$ and $w(I) = (I_R - I_L)/2$. An interval I can also defined by its mean $m(I)$ and width $w(I)$ as $< m(I), w(I) >$, i.e., $I = [I_L, I_R] \equiv < m(I), w(I) >$. Clearly, α-cut of a fuzzy number with continuous membership function can be treated as an interval.

Arithmetic of Interval: Let $\star \in \{+, -, ., /\}$ be a binary operation on the set of positive real numbers. If $A = [A_L, A_R]$ and $B = [B_L, B_R]$ are two closed intervals, then according to Moore [128]:

$$A \star B = \{x \star y : x \in A, y \in B\}$$

Some important arithmetic operations on the closed intervals are summarized as below:

$$
\begin{aligned}
A+B &= [A_L, A_R] + [B_L, B_R] = [A_L + B_L, A_R + B_R] \\
A-B &= [A_L, A_R] - [B_L, B_R] = [A_L - B_R, A_R - B_L] \\
A.B &= [A_L, A_R].[B_L, B_R] \\
&= \Big[Min\{A_L B_L, A_L B_R, A_R B_L, A_R B_R\}, Max\{A_L B_L, A_L B_R, A_R B_L, A_R B_R\}\Big] \\
A/B &= [A_L, A_R]/[B_L, B_R] = [A_L, A_R].\Big[\frac{1}{B_R}, \frac{1}{A_R}\Big], \text{ where } 0 \notin B \\
kA &= \begin{cases} [kA_L, kA_R], \text{ for } k \geq 0 \\ [kA_R, kA_L], \text{ for } k < 0, \text{ where } k \text{ is a real number.} \end{cases}
\end{aligned}
$$

2.1.5 Ranking of intervals

Several approaches are proposed by different researchers for ordering intervals. Sengupta and Pal [170] presented a detailed comparison of different approaches together with their merits and demerits. In this thesis, the interval comparison approach, Fuzzy Preference Ordering (FPO) of intervals, made by Sengupta and Pal [170] is used and discussed below.

Fuzzy Preference Ordering (FPO) of intervals: According to Sengupta and Pal [170], the interval ordering scheme, FPO, is the most efficient interval ranking approach. They consider the following trivial assumptions for any maximization problem involving interval objectives.

1. More profit is better than less profit.
2. More certainty is better than less certainty.
3. If more profit is associated with more uncertainty, a DM undergoes a trade-off between the two.
4. To a pessimistic DM, assumption 2 is somewhat more important than assumption 1 (Obviously to an optimistic DM, assumption 1 is somewhat more important than assumption 2).

Following these assumptions, when profits are represented by intervals, they ordered any pair of profit intervals A and B, as (A, B) or (B, A) according as

$m(A) \leq m(B)$ or $m(A) \geq m(B)$ and then classified the order pair of intervals into two sets S_1 and S_2 as follows:

1. $S_1 = \{(A, B) | m(A) \leq m(B), w(A) \geq w(B)\}$
2. $S_2 = \{(A, B) | m(A) \leq m(B), w(A) \leq w(B)\}$

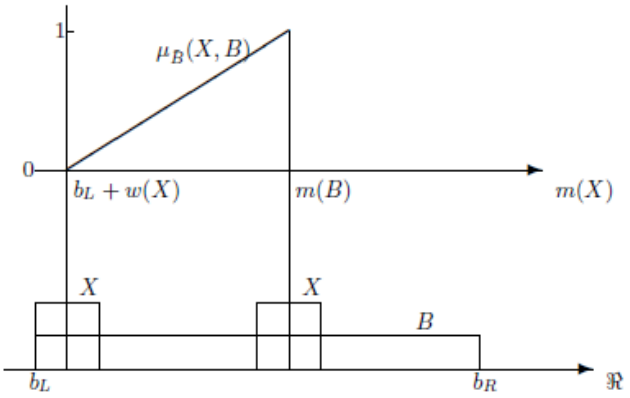

FIGURE 2.10: Membership function of rejection of B for the pair of intervals (X,B) in S_2

Then for a maximization problem, for any two profit intervals A, B, any one of the following two cases may arises assuming $m(A) \leq m(B)$:

- For $(A, B) \in S_1$ unless A and B are identical, B is always the best choice. Here $m(A) \leq m(B)$, i.e., mean profit due to B is greater than mean profit due to A. Moreover $w(A) \geq w(B)$ implies uncertainty in the choice of A is more than that of B. So obviously B is best choice.
- If $(A, B) \in S_2$ then DM is in dilemma. In this case though mean profit due to B is better than that of A, uncertainty in the choice of A is more than that of B. So there is a trade off between uncertainty and mean profit and hence indifference between A and B occurs. According to Sengupta and Pal [170] depending upon the order of priority between mean and width, fuzzy preference between A and B may be constructed to resolve the case of indifference between A and B.

In order to develop fuzzy preference between the indifference pair (X, B) in S_2, a fuzzy set $R(B)$ as rejection of B compared to X in S_2 is defined as

$$R(B) = \{(X, B) \in S_2 | X = [X_L, X_R], m(X) \leq m(B),\ w(X) < w(B)\}$$

with membership function $\mu_{R(B)}(X, B)$ ($\mu_{R(B)}$ being a function $S_2 \to [0, 1]$), given by (cf. Figure 2.10)

$$\mu_{R(B)}(X, B) = \begin{cases} 1 & \text{if } m(X) = m(B) \\ \max\left\{0, \frac{m(X)-(B_L+w(X))}{m(B)-(B_L+w(X))}\right\} & \text{if } m(B) \geq m(X) \geq (B_L + w(X)) \\ 0 & \text{otherwise.} \end{cases}$$

Clearly membership function depends on the mean values and uncertainty (width) of the intervals. According to the above definition following conclusions are obvious:

- If $\mu_{R(B)}(X, B) = 1$, B is definitely rejected compare to X. This is because in this case $m(X) = m(B)$ but $w(X) < w(B)$, i.e., though mean profit in both the cases is same uncertainty in B is more compared to X and hence X is accepted.

- If $\mu_{R(B)}(X, B) = 0$, B is definitely accepted compared to X, as in this case $X_L < B_L$, $X_R < B_R$ together with $m(X) < m(B)$.

- If $\mu_{R(B)}(X, B) \in (0, 1)$, B is accepted/rejected according to DM's preference. If $\mu_{R(B)}(X, B)$ is nearly 1, $m(X)$ is nearly equal to $m(B)$ moreover uncertainty in B compared to X is high and hence B may be rejected. But, if $\mu_{R(B)}(X, B)$ is nearly 0, X_L is nearly equal to B_L. Moreover $m(X) < m(B)$, $w(X) < w(B)$, so X contained in the lower part of B. Clearly, in this case profit due to B will not be worsen than X in any situation and hence B should be accepted compared to X. For other values of $\mu_{R(B)}(X, B)$ rejection of B compared to X depends upon the DM's choice. Optimistic DM will reject B compared to X for large values of $\mu_{R(B)}(X, B)$ only but, a pessimistic DM will reject B compared to X for small values of $\mu_{R(B)}(X, B)$. To solve the proposed model, it is assumed that if $\mu_{R(B)}(X, B) \geq 0.5$, B is rejected compared to X. Here B is rejected compared to X if $X_L + \alpha w(X) \geq B_L + \alpha w(B)$, where $\alpha = 0.5$, i.e., if $X_L + m(X) \geq B_L + m(B)$.

2.2 Solution Methods/Techniques in Crisp Environment

Any real life problem involves some objectives and normally there exists some restrictions (constraints) which prevents to reach the objective goal. The decision maker needs proper decision to face the problem, normally called optimal decision. The decisions may be logical or numeric. For numeric decisions, the problem involves some decision variables (real valued) and some objectives (functions of the decision variables) which have to be optimized (minimized or maximized) under some constraints. The problem involving only one such objective under some constraints is known as single objective optimization problem (SOOP) as presented below:

$$\left.\begin{array}{l} \text{Find} \quad x = (x_1, x_2, ..., x_n)^T \\ \text{which maximizes/minimizes } f(x) \\ \text{subject to} \quad x \in X \\ \text{where } X = \left\{ x : \begin{array}{ll} g_i(x) \le 0, & i = 1, 2, ..., l \\ h_j(x) = 0, & j = 1, 2, ..., m \\ x_k \ge 0, & k = 1, 2, ..., n \end{array} \right\} \end{array}\right\} \tag{2.27}$$

where, $f(x)$, $g_i(x)$, $i = 1, 2, ..., l$ and $h_j(x)$, $j = 1, 2, ..., m$ are functions defined on n-dimensional set.

It is noted that, when both the objective function and constraints are linear, the above SOOP becomes a single objective linear optimization problem (SOLOP). Otherwise, it is a single objective non-linear optimization problem (SONLOP).

A decision variable vector x satisfying all the constraints is called a feasible solution to the problem. The collection of all such solutions forms a feasible region. The SOOP (2.27) is to find a feasible solution x^* such that for each feasible point x, $f(x) \le f(x^*)$ for maximization problem and $f(x) \ge f(x^*)$ for minimization problem. Here, x^* is called an optimal solution or solution to the problem.

Local Minimum: $x^* \in X$ is said to be a local minima of (2.27), if there exists an $\epsilon > 0$ such that $f(x) \ge f(x^*)$, $\forall x \in X : |x - x^*| < \epsilon$.

Convex Function: A function $f(x_1, x_2, ..., x_n)$ is convex, if the Hessian Matrix, given by $H(x_1, x_2, ..., x_n) = \left[\frac{\partial^2 f}{\partial x_i \partial x_j}\right]_{n \times n}$, is positive semi-definite/positive definite.

Global Minimum: $x^* \in X$ is said to be a global minima of (2.27), if $f(x) \geq f(x^*),\ \forall x \in X$. Otherwise, if the function $f(x)$ is convex then the local minimum solution $x \in X$ is global minimum.

Convex Programming Problem: The problem defined in (2.27) is said to be convex programming problem, if the objective function $f(x_1, x_2, ..., x_n)$ and the constraint functions $g_i(x_1, x_2, ..., x_n),\ i = 1, 2, ..., m$ are convex.

For the solution of SONLOP by any available non-linear programming (NLP) method, local optimal solutions are guaranteed. Also, it is known that, a local minimum/maximum solution is a global minimum/maximum for a convex/concave optimization (i.e., a NLP problem to minimize a convex function or to maximize a concave function) problem.

Lot of mathematical techniques based on linearization, gradient based, evolutionary algorithms, stochastic search algorithms etc., are available in the literature to solve such type of SONLOP. Here, few methods are illustrated, which have been used in this thesis to solve the inventory problems, non-linear in nature.

Necessary Condition for Optimality: If a function $f(x)$ is defined for all $x \in X$ and has a relative minimum at $x = x^*$, where $x^* \in X$ and all the partial derivatives $\frac{\partial f(x)}{\partial x_r}$ for $r = 1, 2, ..., n$ are exists at $x = x^*$, then $\frac{\partial f(x^*)}{\partial x_r} = 0$.

Sufficient Condition for Optimality: The sufficient condition for a stationary point x^* to be an extreme point is that the matrix of second order partial derivatives (Hessian Matrix) of $f(x)$ evaluated at $x = x^*$ is (i) positive definite, when x^* is a relative minimum point, and (ii) negative definite, when x^* is a relatively maximum point.

For the solution of a SONLOP, there exists a lot of mathematical techniques in the literature based on linearisation, use of gradient etc. But these techniques are not suitable for searching solutions of real life decision making problems involving several variables and constraints in a reasonable time window. Moreover, these techniques are not capable to find the solutions of the optimisation problems in imprecise environments. To overcome these situations, the researchers develop various heuristic search algorithms, meta heuristic search algorithms, stochastic search algorithms etc., which are available in the literature. In this thesis, to solve the continuous non-linear optimization problems some soft computing approaches/

techniques have been proposed/developed; their efficiency in solving the real life SOOPs are tested and illustrated.

2.2.1 Classical Optimization Technique

The classical optimization techniques are very useful to obtain optimal solution of the problems involving continuous and differentiable functions. This type of techniques are analytical in nature and hence the exact results can be found. Maximum and minimum points for unconstrained and constrained continuous objective functions can be obtained using these techniques. These techniques have some limitations. The problems in crisp environment can be solved using these techniques, but the problems in imprecise environment (fuzzy, rough etc.) can not be solved. One of these methods is discussed below:

2.2.1.1 Generalized Reduced Gradient (GRG) Technique

The GRG technique is a method for solving NLP problems for handling equality as well as inequality constraints. Consider the NLP problem:

$$\left.\begin{array}{l} \text{Find} \quad x = (x_1, x_2, \ldots x_n)^T \\ \text{which maximizes } f(x) \\ \text{subject to} \quad x \in X \\ \text{where } X = \left\{ x : \begin{array}{ll} g_i(x) \le 0, & i = 1, 2, \ldots, l \\ h_j(x) = 0, & j = 1, 2, \ldots, m \\ x_k \ge 0, & k = 1, 2, \ldots, n \end{array} \right\} \end{array}\right\} \tag{2.28}$$

By adding a non-negative slack variable $s_i\,(\ge 0)$, $i = 1, 2, \ldots, l$ to each of the above inequality constraints, the problem (2.28) can be stated as

$$\left.\begin{array}{l} \text{Maximize} \quad f(x) \\ \text{subject to} \quad x \in X \\ \text{where } X = \left\{ x : \begin{array}{ll} x = (x_1, x_2, \ldots, x_n)^T & \\ g_i(x) + s_i = 0, & i = 1, 2, \ldots, l \\ h_j(x) = 0, & j = 1, 2, \ldots, m \\ x_k \ge 0, & k = 1, 2, \ldots, n \\ s_i \ge 0, & i = 1, 2, \ldots, l \end{array} \right\} \end{array}\right\} \tag{2.29}$$

where, the lower and the upper bounds on the slack variables, $s_i,\ i = 1, 2, ..., l$ are taken as a zero and a large number (infinity) respectively.

Denoting s_i by x_{n+i}, $g_i(x) + s_i$ by ξ_i, $h_j(x)$ by ξ_{l+j}, the above problem can be rewritten as

$$\left.\begin{array}{l} \text{Maximize} \quad f(x) \\ \text{subject to} \quad x \in X \\ \text{where } X = \left\{ x : \begin{array}{ll} x = (x_1, x_2, ..., x_{n+l})^T & \\ \xi_i(x) = 0, & i = 1, 2, ..., l+m \\ x_k \geq 0, & k = 1, 2, ..., n+l \end{array} \right\} \end{array}\right\} \tag{2.30}$$

This GRG technique is based on the idea of elimination of variables using the equality constraints. Theoretically, $(l+m)$ variables (dependent variables) can be expressed in terms of remaining $(n-m)$ variables (independent variables). Thus one can divide the $(n+l)$ decision variables arbitrarily into two sets as

$$x = (y, z)^T$$

where, y is $(n-m)$ design or independent variables and z is $(l+m)$ state or dependent variables and

$$\begin{aligned} y &= (y_1, y_2, ..., y_{n-m})^T \\ z &= (z_1, z_2, ..., z_{l+m})^T \end{aligned}$$

Here, the design variables are completely independent and the state variables are dependent on the design variables used to satisfy the constraints

$$\xi_i(x) = 0,\ i = 1, 2, ..., l+m$$

Consider the first variations of the objective and constraint functions:

$$df(x) = \sum_{i=1}^{n-m} \frac{\partial f}{\partial y_i} dy_i + \sum_{i=1}^{l+m} \frac{\partial f}{\partial z_i} dz_i = \nabla_y^T f\, dy + \nabla_z^T f\, dz \tag{2.31}$$

$$\begin{aligned} d\xi_i(x) &= \sum_{j=1}^{n-m} \frac{\partial \xi_i}{\partial y_j} dy_j + \sum_{j=1}^{l+m} \frac{\partial \xi_i}{\partial z_j} dz_j \\ \text{or} \quad d\xi &= C\, dy + D\, dz \end{aligned} \tag{2.32}$$

where, $\nabla_y^T f = \left(\frac{\partial f}{\partial y_1}, \frac{\partial f}{\partial y_2}, ..., \frac{\partial f}{\partial y_{n-m}} \right)$ and $\nabla_z^T f = \left(\frac{\partial f}{\partial z_1}, \frac{\partial f}{\partial z_2}, ..., \frac{\partial f}{\partial z_{l+m}} \right)$.

$$C = \begin{bmatrix} \frac{\partial \xi_1}{\partial y_1} & \cdots & \cdots & \cdots & \frac{\partial \xi_1}{\partial y_{n-m}} \\ \frac{\partial \xi_2}{\partial y_1} & \cdots & \cdots & \cdots & \frac{\partial \xi_2}{\partial y_{n-m}} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \frac{\partial \xi_{l+m}}{\partial y_1} & \cdots & \cdots & \cdots & \frac{\partial \xi_{l+m}}{\partial y_{n-m}} \end{bmatrix}, \qquad D = \begin{bmatrix} \frac{\partial \xi_1}{\partial z_1} & \cdots & \cdots & \cdots & \frac{\partial \xi_1}{\partial z_{l+m}} \\ \frac{\partial \xi_2}{\partial z_1} & \cdots & \cdots & \cdots & \frac{\partial \xi_2}{\partial z_{l+m}} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \frac{\partial \xi_{l+m}}{\partial z_1} & \cdots & \cdots & \cdots & \frac{\partial \xi_{l+m}}{\partial z_{l+m}} \end{bmatrix},$$

$$\begin{aligned} dy &= (dy_1, dy_2, ..., dy_{n-m})^T \\ \text{and} \quad dz &= (dz_1, dz_2, ..., dz_{l+m})^T \end{aligned}$$

Assuming that the constraints are originally satisfied at the vector x $(\xi(x) = 0)$, any change in the vector dx must correspond to $d\xi = 0$ to maintain feasibility at $x + dx$. Thus, the Eqn. (2.32) can be solved as

$$\begin{aligned} & Cdy + Ddz = 0 \\ \text{or} \qquad & dz = -D^{-1}Cdy \end{aligned} \tag{2.33}$$

The change in the objective function due to the change in x is given by the Eqn. (2.31), which can be expressed, using Eqn. (2.33) as

$$\begin{aligned} df(x) &= (\nabla_y^T f - \nabla_z^T f D^{-1} C) dy \\ \text{or} \quad \frac{df(x)}{dy} &= G_R \\ \text{where,} \quad G_R &= \nabla_y^T f - \nabla_z^T f D^{-1} C \end{aligned}$$

is called the *generalized reduced gradient*. Geometrically, the reduced gradient can be described as a projection of the original n-dimensional gradient into the $(n-l)$ dimensional feasible region described by the design variables.

A necessary condition for the existence of minimum of an unconstrained function is that the components of the gradient vanish. Similarly, a constrained function assumes its minimum value when the appropriate components of the reduced gradient are zero. In fact, the reduced gradient G_R can be used to generate a

search direction S to reduce the value of the constrained objective function. Similarly, to the gradient ∇f that can be used to generate a search direction S for an unconstrained function. A suitable step length λ is to be chosen to minimize the value of $f(x)$ along the search direction. For any specific value of λ, the dependent variable vector z is updated using Eqn. (2.33). Noting that the Eqn. (2.32) is based on using a linear approximation to the original non-linear problem, so the constraints may not be exactly equal to zero at λ, i.e., $d\xi \neq 0$. Hence, when y is held fixed, in order to have

$$\xi_i(x) + d\xi_i(x) = 0, \quad i = 1, 2, ..., l + m \tag{2.34}$$

following must be satisfied.

$$\xi(x) + d\xi(x) = 0 \tag{2.35}$$

Using Eqn. (2.32) for $d\xi$ in Eqn. (2.35), following is obtained

$$dz = D^{-1}(-\xi(x) - Cdy) \tag{2.36}$$

The value dz given by Eqn. (2.36) is used to update the value of z as

$$z_{update} = z_{current} + dz \tag{2.37}$$

The constraints evaluated at the updated vector x, and the procedure of finding dz using Eqn. (2.37) is repeated until dz is sufficiently small.

2.2.2 Soft Computing Techniques

Nowadays to solve the complicated real life problems, the researchers are attracted by different heuristic optimization techniques, which provide the efficient and stable solution to the problems. Among the basic heuristic approaches, Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) are mostly used for different optimisation problems of science and technology due to the generality of the algorithms [10, 51, 70, 89]. During the last decade different inventory models involving several variables and constraints are solved using various heuristic

techniques by several researchers [52, 108, 112]. Also the stochastic search algorithms such as Simulated Annealing (SA) provide an efficient feasible solution to the complex problems [111]. Recently, Karaboga [90] developed a meta heuristic mimicking the foraging behaviour/nature of honey bees and named as Artificial Bee Colony (ABC) algorithm. From the literature, it is observed that the algorithm is not only efficient it also provide the solution upto a desired number of decimal presentation. In this thesis, for the optimization purpose, some well known heuristic algorithms have been implemented and also some hybrid heuristic/meta heuristic approaches have been developed and implemented. These are briefly discussed below.

2.2.2.1 Particle Swarm Optimization (PSO) Technique

Particle Swarm Optimization (PSO) is a heuristic search algorithm was developed by Kennedy and Eberhart [92] in mid-nineties of the last century by mimicking the natural behaviour of a folk of birds searching for their food sources. A folk of birds normally search for their food sources depending upon their own experiences and the best experience among the birds. Same phenomenon is mimicked to create a PSO. The algorithm is suitable for maximization as well as minimization problems. Consider such an optimization problem having objective function $f(X)$ and decision vector $X = (x_1, x_2, ..., x_n)$. In a PSO [53] algorithm, optimal solution is analogous to the food source, position of a bird is compared with a potential solution, velocity of a bird is used to find its neighbour position, i.e., neighbour (perturbed) solution. As folk of birds together search for their food sources and position of a bird is compared with a potential solution, a set of randomly generated potential solutions $P(0)$ is used in the algorithm for searching the optimal solution and is called initial swarm. Let $P(0) = \{X_1(0), X_2(0), ..., X_N(0)\}$, i.e., swarm consists of N potential solutions $X_1(0)$, $X_2(0)$, ..., $X_N(0)$. Here 0 represent iteration number, t, i.e., initially $t = 0$. Each solution (bird/particle) $X_i(t)$ has a velocity of movement $V_i(t)$, where $V_i(0)$ is initialised with some value. Each particle $X_i(t)$ has also its own best position of movement $X_{besti}(t)$. Best position found by the swarm is stored in another vector $X_{gbest}(t)$. In each iteration t, position of a particle $X_i(t)$ and velocity $V_i(t)$ are updated to $X_i(t+1)$ and $V_i(t+1)$ using $X_{besti}(t)$, $X_{gbest}(t)$, $X_i(t)$ and $V_i(t)$. The rules used for the purpose are as

follows:

$$V_i(t+1) = wV_i(t) + \mu_1 r_1(X_{pbesti}(t) - X_i(t)) + \mu_2 r_2(X_{gbest}(t) - X_i(t)) \quad (2.38)$$

$$X_i(t+1) = X_i(t) + V_i(t+1) \quad (2.39)$$

The parameter ω $(0 < \omega < 1)$, known as the inertia weight which controls the influence of previous velocity on the new velocity. Here, the value of ω is taken as 0.7298. The parameters μ_1 and μ_2 are set to the constant values, which are normally taken as 2. In this thesis, the values of μ_1 and μ_2 are considered as 1.49618. r_1 and r_2 are two random values uniformly distributed in $[0, 1]$. The algorithm for the implementation of the optimization technique is given below, where it is assumed that N is the number of particles. V_{max} represent the maximum velocity of a particle, $B_{il}(t)$ and $B_{iu}(t)$ represent the lower boundary and the upper boundary of i-th variable respectively. check_constraint function check whether the solution $X_i(t)$ satisfies the constraints of the problem or not. It returns 1 if the solution $X_i(t)$ satisfies the constraints of the problem, otherwise it returns 0.

PSO Algorithm:

1. Initialize w, μ_1, μ_2, N, V_{max} and $Maxgen$.
2. Set iteration counter $t = 0$ and randomly generate initial swarm $P(t)$ of N particles (solutions).
3. Determine objective value of each solution $X_i(t)$ and find $X_{gbest}(t)$ using dominance property.
4. Set initial velocity $V_i(t)$, $\forall X_i(t) \in P(t)$ and set $X_{pbesti}(t) = X_i(t)$, $\forall X_i(t) \in P(t)$.
5. While $(t < Maxgen\ )$ do
6. For $i = 1 : N$ do
7. $V_i(t+1) = wV_i(t) + \mu_1 r_1(X_{pbesti}(t) - X_i(t)) + \mu_2 r_2(X_{gbest}(t) - X_i(t))$
8. If $(V_i(t+1) > V_{max})$, then set $V_i(t+1) = V_{max}$
9. If $(V_i(t+1) < -V_{max})$, then set $V_i(t+1) = -V_{max}$
10. $X_i(t+1) = X_i(t) + V_i(t+1)$
11. If $(X_i(t+1) > B_{iu}(t))$, then set $X_i(t+1) = B_{iu}(t)$
12. If $(X_i(t+1) < B_{il}(t))$, then set $X_i(t+1) = B_{il}(t)$
13. If check_constraint $(X_i(t+1)) = 0$
14. Set $X_i(t+1) = X_i(t)$, $V_i(t+1) = V_i(t)$
15. Else

16. If $X_i(t+1)$ dominates $X_{pbesti}(t)$, then set $X_{pbesti}(t+1) = X_i(t+1)$
17. If $X_i(t+1)$ dominates $X_{gbest}(t)$, then set $X_{gbest}(t+1) = X_i(t+1)$
18. End If
19. End For
20. Set $t = t+1$
21. End While
22. Output: $X_{gbest}(t)$
23. End Algorithm

Different procedures of the PSO:

(a) Representation of solutions: A n-dimensional real vector, $X_i = (x_{i1}, x_{i2}, ..., x_{in})$, is used to represent i-th solution, where x_{i1}, x_{i2}, ... , x_{in} represent n decision variables of the decision making problem under consideration.

(b) Initialization: N such solutions $X_i = (x_{i1}, x_{i2}, ...x_{in})$, $i = 1, 2, ..., N$, are randomly generated by random number generator within the boundaries of each variable $[B_{jl}, B_{ju}]$, $j = 1, 2, ..., n$. Initialize $(P(0))$ subfunction is used for this purpose.

(c) Dominance property: For crisp maximization problem, a solution X_i dominates a solution X_j, if the objective value due to the the solution X_i is grater than that of X_j. Using this dominance property, PSO can be used to optimize optimization problem with crisp as well as imprecise (fuzzy, rough etc.) objective.

For fuzzy objective, let $\tilde{Z}_i$ and $\tilde{Z}_j$ be the objective values due to the solutions X_i and X_j respectively. Then X_i dominates X_j, if $Cr(\tilde{Z}_i > \tilde{Z}_j) > 0.5$. This is a valid comparison operator as $Cr(\tilde{Z}_i > \tilde{Z}_j) + Cr(\tilde{Z}_i \leq \tilde{Z}_j) = 1$. Here credibility of an event is made by the formula (2.7).

For rough objective, let $\check{Z}_i$ and $\check{Z}_j$ be the objective values due to the solutions X_i and X_j respectively. Then X_i dominates X_j, if $Tr(\check{Z}_i > \check{Z}_j) > 0.5$. This is a valid comparison operator as $Tr(\check{Z}_i > \check{Z}_j) + Tr(\check{Z}_i \leq \check{Z}_j) = 1$. Here trust measure of an event is made by the formula (2.24).

(d) Parameter setting and Implementation: With the above function and values, the algorithm is implemented using C-programming language. Different parametric values of the algorithm used to solve some models of this dissertation are set using Taguchi approach [204] (cf. § 2.2.2.2) and Engelbrecht [53] as follows: $N = 20$, $V_{max} = 0.15$, $\mu_1 = 1.49618$, $\mu_2 = 1.49618$, $w = 0.7298$, $Maxgen = 500$.

To test the performance of the proposed PSO, it is tested against a set of benchmark test problems available in the literature and are listed below. The results of the test function are presented in Table 2.1. From Table 2.1, it is clear that the implemented PSO is efficient enough to solve continuous optimisation problems.

List of Test Functions (TF)

TF-1: (Taken from [11]):
$SH(x_1, x_2) = \sum_{j=1}^{5} j \times cos[(j+1) \times x_1 + j] \times \sum_{j=1}^{5} j \times cos[(j+1) \times x_2 + j]$,
$-10 \leq x_1, x_2 \leq 10$. This problem has 760 local minima and 18 global minima. At global minima (x_1, x_2), $SH(x_1, x_2) = -186.7309$.

TF-2: (Taken from [11]):
$MZ(x_1, x_2, ..., x_n) = -\sum_{i=1}^{n} sin(x_i).[sin(i.(x_i)^2/\pi)]^{2m}$, $-\pi \leq x_1, x_2, ..., x_n \leq \pi$, where $m = 10$. For $n = 2$, it has one global minima at $(x_1, x_2) = (2.25, 1.57)$ and $MZ(2.25, 1.57) = -1.80$.

TF-3: (Taken from [11]):
$F2(x_1, x_2) = 100 \times (x_2^2 - x_1) + (1 - x_1)$, $-2.048 \leq x_1, x_2 \leq 2.048$. It has one minima at $(x_1, x_2) = (2.048, 0)$ and $F2(2.048, 0) = -205.8480$.

TF-4: (Taken from [11]):
$DJ(x_1, x_2, x_3) = x_1^2 + x_2^2 + x_3^2$, $-5.12 \leq x_1, x_2, x_3 \leq 5.12$.
It has one global minima at $(x_1, x_2, x_3) = (0, 0, 0)$ and $DJ(0, 0, 0) = 0$.

TF-5: (Taken from [123]):
$F(x_1, x_2) = 100(x_2 - x_1^2)^2 + (x_1 - 1)^2$, such that $x_1 + x_2^2 \geq 0$, $x_1^2 + x_2 \geq 0$, $-0.5 \leq x_1 \leq 0.5$, $-1.0 \leq x_2 \leq 1.0$.
It has one global minima at $(x_1, x_2) = (0.5, 0.25)$ and $F(0.5, 0.25) = 0.25$.

2.2.2.2 Tuning of PSO parameter using Taguchi method

To solve the problems of some models of the thesis, PSO technique is used. In PSO, some parameters are used which are swarm size (N), velocity (V_{max}) for the movement of the solution, some constants (μ_1, μ_2, w). If the values of these parameters are changed, then the number of function evaluations vary. So, there is need to set (tuning) the parameters so that the number of function evaluations is minimized. To tuning these parameters, Taguchi method [204] is used.

TABLE 2.1: Results of Test Functions following PSO Approach

TF	Results Obtained	% of Success for 50 Runs	Average Error
TF-1	$SH(x^*) = -186.7309$	96	0.022040
TF-2	$MZ(2.2029, 1.5708) = -1.8013$	68	0.256416
TF-3	$F2(2.048, -0.0002) = -205.8480$	100	0
TF-4	$DJ(0, 0, 0) = 0$	100	0
TF-5	$F(0.5, 0.25) = 0.25$	100	0

For TF-1, 18 global minima (obtained by PSO approach) are
$x^* = [(-7.0835, 4.8581), (4.8581, 5.4828), (-7.7083, -0.8003), (5.4829, -7.7083), (-7.0835, -1.4251), (-1.4251, -7.0835), (-0.8004, -1.4251), (4.8580, -0.8003), (-0.8004, 4.8580), (5.4829, 4.8581), (5.4829, -1.4251), (-7.0835, -7.7083), (4.8581, -7.0835), (-1.4251, 5.4829), (-0.8003, -7.7084), (-1.4251, -0.8003), (-7.7083, -7.0835), (-7.7084, 5.4829)]$

TABLE 2.2: PSO parameters and their levels

Symbol	PSO parameter	Level-1	Level-2	Level-3
A	swarm size (N)	20	30	40
B	velocity (V_{max})	0.05	0.10	0.15
C	constant (μ_1=μ_2)	1.45	1.49618	1.55
D	constant (w)	0.71	0.7298	0.75

TABLE 2.3: L_9 orthogonal array

Run	PSO parameter			
	A	B	C	D
	swarm size (N)	velocity (V_{max})	constant (μ_1=μ_2)	constant (w)
1	1	1	1	1
2	1	2	2	2
3	1	3	3	3
4	2	1	3	2
5	2	2	1	3
6	2	3	2	1
7	3	1	2	3
8	3	2	3	1
9	3	3	1	2

In Taguchi method, there is need to set the levels of the parameters of PSO at first, which are shown in Table 2.2. A three-level PSO parameter counts for two degrees of freedom. Therefore, there are total eight degrees of freedom for the four PSO parameters. The degrees of freedom for the orthogonal array should be greater than or at least equal to those for the design parameters. So, in the present study, an L_9 orthogonal array (cf. Table 2.3) is selected. This array has four columns and nine rows. The degrees of freedom of this array is eight. Each PSO parameter is assigned to a column and nine combinations of the parameters are required for L_9 orthogonal array.

TABLE 2.4: Required number of function evaluation in different seeds and S/N ratio

Run	Symbol				Function Evaluation			S/N ratio
	A	B	C	D	Seed-1	Seed-2	Seed-3	
1	20	0.05	1.45	0.71	1260	1320	1700	-63.17
2	20	0.10	1.49618	0.7298	640	900	1080	-59.01
3	20	0.15	1.55	0.75	940	1000	920	-59.59
4	30	0.05	1.55	0.7298	1380	1200	2310	-64.61
5	30	0.10	1.45	0.75	1080	1200	1290	-61.53
6	30	0.15	1.49618	0.71	1230	960	1110	-60.87
7	40	0.05	1.49618	0.75	1800	2040	2280	-66.23
8	40	0.10	1.55	0.71	2080	1520	1840	-65.24
9	40	0.15	1.45	0.7298	880	1000	1600	-61.60

For the above combinations of the parameters, the number of function evaluations to find optimal decision in different seeds of the random number generator are tabulated in Table 2.4 and also these numbers are transformed into a signal-to-noise (S/N) ratio by the following formula.

$$\eta = -10log(M.S.D.) \tag{2.40}$$

where, M.S.D. is the mean square deviation for the output characteristic.

There are three types of quality characteristic: the-lower-the-better, the-higher-the-better and the-nominal-the-better [204]. Here, the number of function evaluation is the-lower-the-better quality characteristic. The M.S.D. for this characteristic is calculated by the following formula.

$$M.S.D. = \frac{1}{m}\sum_{i=1}^{m} S_i^2 \tag{2.41}$$

where, S_i is the value of the number of function evaluation for the i-th test.

The mean S/N ratio for each level of each parameter is summarized and called the S/N response table for the number of function evaluation (cf. Table 2.5). For example, the mean S/N ratio for the parameter swarm size (N) at levels 1, 2 and 3 can be calculated by averaging the S/N ratios for the run {1,2,3}, {4,5,6} and {7,8,9} respectively and so on for the other parameters.

Finally, the ANOVA test is done to investigate which design parameters significantly affect the quality characteristic. From Table 2.6, it is observed that the

TABLE 2.5: S/N response table for function evaluation

Symbol	PSO parameter	Mean S/N ratio			
		Level-1	Level-2	Level-3	(Max - Min)
A	swarm size (N)	-60.59	-62.34	-64.36	3.77
B	velocity (V_{max})	-64.67	-61.93	-60.69	3.98
C	constant (μ_1=μ_2)	-63.09	-61.74	-62.45	1.35
D	constant (w)	-62.10	-62.04	-63.15	1.11

TABLE 2.6: Results of the ANOVA for function evaluation

Symbol	PSO parameter	df	SS_B	MS_B	SS_W	MS_W	F-ratio
A	swarm size (N)	2	21.36	10.68	30.01	5.00	2.13
B	velocity (V_{max})	2	24.88	12.44	26.40	4.40	2.83
C	constant (μ_1=μ_2)	2	2.74	1.37	48.58	8.10	0.17
D	constant (w)	2	2.34	1.17	49.00	8.17	0.14

values of F-ratio of each parameter is less than the value of $F(2,6) = 5.14$. So, the change of the parameters are insignificant on the quality characteristic. Based on the S/N and ANOVA analyses, the optimal PSO parameters for the number of function evaluation are the swarm size (N) at level-1, the velocity (V_{max}) at level-3, the constants ($\mu_1 = \mu_2$) at level-2 and the constant (w) at level-2. Due to this reason, this parametric set is used to find marketing decision of the models.

2.2.2.3 Multi-choice Artificial Bee Colony (MCABC) Algorithm

Mimicking the foraging behaviors of honey bee colonies, ABC algorithm was introduced by Karaboga [90] for solving continuous optimization problems. During last decade it has been modified by several researchers to improve its performance and it's different variants are available in the literature [82, 96, 211]. An ABC algorithm normally starts with a set of randomly generated potential solutions in the search space of the optimization problem under consideration. A solution is treated as a food source and the optimal solution is the best food source. It is an iterative search approach, where each iteration consists of three phases – employee bee phase, onlooker bee phase and scout bee phase. Each food source (potential solution) is associated with an employee bee, i.e., the number of solutions equal to the number of employee bees. Each employee bee tries to find a better food source, i.e., tries to find a better neighbour solution using some search strategies. After operations of all the employee bees, i.e., after employee bee phase, onlooker bee phase starts. Number of onlooker bees is fixed and each bee chooses a solution

(food source) from the solution set according to their fitness and tries to find a better neighbour solution using some search strategies. After operation of each onlooker bee, scout bee phase starts. Each solution is associated with a counter which keeps record of the consecutive number of iterations in which the solution does not move (improved). If value of this counter exceeds a predefined fixed limit, then it is regenerated in the search space. The algorithm keeps record of the best solution find so far and after end of the maximum limit of the iterations, the best solution is produced as output. In basic ABC algorithm, only one search strategy was used by the employee bees and the onlooker bees. In this study, a set of search strategies (updating rules) is suggested and a bee chooses a strategy (updating rule) from this set according to the performance of the strategy. The ABC algorithm with the proposed rules is named as Multi-choice Artificial Bee Colony (MCABC) Algorithm and is briefly discussed below.

Let $S = \{X_1, X_2, ..., X_n\}$ be the initial swarm consist of n solutions randomly generated in the search space of the optimization problem under consideration and the problem consists of m variables. Then the i-th solution X_i consists of m components and is represented by $X_i = (x_{i1}, x_{i2}, ..., x_{im})$. Best solution found so far is denoted by $X_B = (x_{B1}, x_{B2}, ..., x_{Bm})$. For the movement of a solution, X_i, perturbation is made on a randomly selected component of the solution. Let j-th component x_{ij} of X_i is selected randomly for perturbation. Then X_i perturbed to $X_i' = (x_{i1}, x_{i2}, ..., x_{ij}', ..., x_{im})$ and perturbation is made by selecting a rule from the following set of rules according to the performance of the rule. In the rules, r_1, r_2 are two randomly generated numbers in $(-1, 1)$ and k, l are randomly selected from the set $\{1, 2, ..., n\}$.

$$x_{ij}' = x_{ij} + r_1(x_{Bj} - x_{ij}) \tag{2.42}$$

$$x_{ij}' = x_{ij} + r_1(x_{kj} - x_{ij}) \tag{2.43}$$

$$x_{ij}' = x_{ij} + r_1(x_{Bj} - x_{ij}) + r_2(x_{kj} - x_{ij}) \tag{2.44}$$

$$x_{ij}' = x_{ij} + r_1(x_{Bj} - x_{ij}) + r_2(x_{kj} - x_{lj}) \tag{2.45}$$

Each rule is associated with a counter which is initialized with 1. If p-th rule is selected for perturbation of X_i and perturbed solution X_i' is better than X_i, then X_i is replaced by X_i' and the counter variable corresponding to p-th rule is increased by 1. Selection of a rule is made by Roulette Wheel selection process [93, 123] depending upon the value of its counter variable. This change is incorporated in

TABLE 2.7: Benchmark functions used for comparison

No. of Function	Name	Search Range	Function
F1	Sphere	[-100,100]	$f_1(\vec{X}) = \sum_{i=1}^{n} x_i^2$
F2	Elliptic	[-100,100]	$f_2(\vec{X}) = \sum_{i=1}^{n} (10^6)^{(i-1)/(n-1)} x_i^2$
F3	SumSquares	[-10,10]	$f_3(\vec{X}) = \sum_{i=1}^{n} i x_i^2$
F4	Schwefel 2.22	[-10,10]	$f_4(\vec{X}) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$
F5	Quartic	[-1.28,1.28]	$f_5(\vec{X}) = \sum_{i=1}^{n} i x_i^4$
F6	Rastrigin	[-5.12,5.12]	$f_6(\vec{X}) = \sum_{i=1}^{n} [x_i^2 - 10cos(2\pi x_i) + 10]$
F7	Weierstrass	[-0.5,0.5]	$f_7(\vec{X}) = \sum_{i=1}^{D} (\sum_{k=0}^{k_{max}} [a^k cos(2\pi b^k (x_i + 0.5))]) - D \sum_{k=0}^{k_{max}} [a^k cos(2\pi b^k 0.5)]$; $a = 0.5,\ b = 3,\ k_{max} = 20$
F8	Rosenbrock	[-10,10]	$f_8(\vec{X}) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$
F9	Penalized 1	[-50,50]	$f_9(\vec{X}) = \frac{\pi}{n} \{10sin^2(\pi y_1) + \sum_{i=1}^{n-1} (y_i - 1)^2 [1 + 10sin^2(\pi y_{i+1})] + (y_n - 1)^2\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$; where, $y_i = 1 + \frac{1}{4}(x_i + 1)$ and $u_{x_i,a,k,m} = \begin{cases} k(x_i - a)^m, x_i > a \\ 0, -a \le x_i \le a \\ k(-x_i - a)^m, x_i < -a \end{cases}$
F10	Penalized 2	[-50,50]	$f_{10}(\vec{X}) = \frac{1}{10} \{sin^2(\pi x_1) + \sum_{i=1}^{n-1} (x_i - 1)^2 [1 + sin^2(3\pi x_{i+1})] + (x_n - 1)^2 [1 + sin^2(2\pi x_n)]\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$
F11	Levy	[-10,10]	$f_{11}(\vec{X}) = \sum_{i=1}^{n-1} (x_i - 1)^2 [1 + sin^2(3\pi x_{i+1})] + sin^2(3\pi x_1) + \lvert x_n - 1 \rvert [1 + sin^2(3\pi x_n)]$
F12	Himmelblau	[-5,5]	$f_{12}(\vec{X}) = \frac{1}{n} \sum_{i=1}^{n} (x_i^4 - 16x_i^2 + 5x_i)$

basic ABC to create MCABC and is used to find marketing decisions of the some models of the thesis.

Implementation and Testing: With the above function and values the algorithm is implemented using Dev C++ (version 5.11) programming language in a computer with core i3 CPU, 2 GB RAM, 2.53 GHz processor speed and Windows-7 operating system.

To test the performance of the proposed MCABC algorithm, it is tested against a set of benchmark test problems available in the literature and are listed in Table 2.7. For each test function, the number of variables is considered as 30, i.e., $n = 30$. The parametric values of the algorithm used for the purpose are taken as: number of employee bees (eb)= 40, number of onlooker bees (ob)= 40 and upper limit of the counter of a solution is $n * eb = 1200$, i.e., if a solution is not moved by the employee bee or by any onlooker bee in 1200 attempts, then the scout bee will regenerate it and its counter is set to 0. The maximum number of iterations used to find the optimal solution is taken as 6000. For each of the test problem, the algorithm is run 10 times with different seeds of random number generator and corresponding mean and standard deviation of the obtained objective values are presented in Table 2.8. Performance of the algorithm is compared with a recently published ABC variant, ABCVSS [96]. In the Table 2.8, results of ABCVSS [96] are taken from the corresponding paper. From the Table 2.8, it is clear that performance of MCABC

TABLE 2.8: Comparison of results of test functions for $n = 30$

Function	ABCVSS		Proposed Algorithm	
	Mean	Standard Deviation	Mean	Standard Deviation
F1	1.53E-81	8.37E-81	0	0
F2	4.82E-82	2.63E-81	0	0
F3	3.19E-89	1.48E-88	0	0
F4	7.89E-43	4.32E-42	0	0
F5	3.25E-154	1.78E-153	0	0
F6	0	0	0	0
F7	0	0	0	0
F8	3.87E-01	1.54E+00	9.21E-03	7.49E-03
F9	1.57E-32	5.57E-48	1.57E-32	2.74E-48
F10	1.35E-32	5.57E-48	1.50E-33	1.71E-49
F11	1.35E-31	6.68E-47	1.35E-31	0
F12	-7.83E+01	3.02E-10	-7.83E+01	1.42E-14

TABLE 2.9: Results of some test functions for different number of iterations

Function	max iteration = 1000		max iteration = 2000		max iteration = 3000		max iteration = 4000	
	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev
F1	2.91E-13	3.81E-13	8.09E-30	2.34E-29	2.66E-46	7.96E-46	3.22E-62	9.65E-62
F2	6.36E-12	7.14E-12	6.34E-30	5.07E-30	1.39E-47	1.22E-47	1.70E-65	3.98E-65
F3	1.42E-14	1.34E-14	1.97E-31	5.15E-31	2.86E-49	7.38E-49	2.74E-67	5.10E-67
F4	1.24E-07	3.78E-08	1.08E-16	1.01E-16	1.13E-25	1.54E-25	8.00E-35	1.64E-34
F5	3.68E-32	6.00E-32	4.61E-65	1.19E-64	4.88E-98	1.20E-97	2.56E-129	7.48E-129
F6	3.40E-01	4.44E-01	3.35E-16	1.00E-15	0	0	0	0
F7	2.72E-04	1.19E-04	0	0	0	0	0	0

is better than ABCVSS. Again according to Kiran et al. [96], performance of ABCVSS is better than the existing other ABC variants. From Table 2.8, it is clear that MCABC produces exact optimal solution for any test problem in most of the runs of the algorithm (mean values of the objectives are 0 or nearly 0 for all the test problems). So the performance of MCABC is acceptable for continuous optimization problems. Also a study is made to check the convergence of the global best solution to the optimal solution for different test problems with respect to the number of iterations and the results are presented in Table 2.9. It is clear from the Table 2.9 that for each of the considered test problems the corresponding global best solution gradually converges to the corresponding optimal solution as iteration increases.

MCABC for Imprecise Environment: MCABC is capable of solving any continuous optimization problem in crisp environment. It can be used to solve

optimization problem in any imprecise environment also. To find the optimal decision, this algorithm needs only comparison of objectives due to two solutions X_1 and X_2. For interval valued objectives it can be done by fuzzy preference ordering of intervals [99, 170]. For fuzzy objectives it can be done by credibility measure approach [150]. For rough objectives it can be done by trust measure approach [150].

- Let $\tilde{F}_1$ and $\tilde{F}_2$ be two fuzzy objectives due to two potential solutions X_1 and X_2 of the fuzzy optimization problem (having fuzzy objective function $\tilde{F}$ and crisp decision vector X) under consideration. Then for maximisation problem $\tilde{F}_1$ is better than $\tilde{F}_2$ if $Cr(\tilde{F}_1 > \tilde{F}_2) > 0.5$, where $Cr(\tilde{F}_1 > \tilde{F}_2)$ represents the credibility measure of the fuzzy event $\tilde{F}_1 > \tilde{F}_2$. It is a valid fuzzy comparison as for any fuzzy event $\tilde{A} > \tilde{B}$, the following relation is always holds [105, 150]

$$Cr(\tilde{A} > \tilde{B}) + Cr(\overline{\tilde{A} > \tilde{B}}) = 1$$

  where, $(\overline{\tilde{A} > \tilde{B}})$ represents the complement of the event $(\tilde{A} > \tilde{B})$, i.e., the event $(\tilde{A} \leq \tilde{B})$.

- Let $\check{F}_1$ and $\check{F}_2$ be two rough objectives due to two potential solutions X_1 and X_2 of the rough optimization problem (having rough objective function $\check{F}$ and crisp decision vector X) under consideration. Then for maximisation problem $\check{F}_1$ is better than $\check{F}_2$ if $Tr(\check{F}_1 > \check{F}_2) > 0.5$, where $Tr(\check{F}_1 > \check{F}_2)$ represents the trust measure of the rough event $\check{F}_1 > \check{F}_2$. It is a valid rough comparison as for any rough event $\check{A} > \check{B}$, the following relation is always holds [105, 150]

$$Tr(\check{A} > \check{B}) + Tr(\overline{\check{A} > \check{B}}) = 1$$

  where, $(\overline{\check{A} > \check{B}})$ represents the complement of the event $(\check{A} > \check{B})$, i.e., the event $(\check{A} \leq \check{B})$.

Constraint Handling: The algorithm can deal with constraint optimization. In the presence of constraint, at the time of generation of a neighbour solution/new solution, constraint checking is done by a separate sub-program. If the solution satisfies the constraint of the problem, then it is included in the solution set, otherwise it is rejected.

2.2.2.4 Mixed-mode Multi-choice Artificial Bee Colony (MMCABC) Algorithm

During last three decades different heuristic techniques play major role in the real life decision making problems in different directions of science and technology. Among different heuristics of continuous optimization, ABC draws more attention during the last decade due to its performance and consistency. Karaboga [90] introduced ABC algorithm by simulating the different phases involved in the search process of foods of the honey bees. It is capable of producing optimal results with any desired degree of accuracy in a reasonable computation time. After his novel work, its different variants are produced by several researchers during the last decade which are successful in the respective fields [18, 91, 132, 177]. Though ABC is successful in continuous optimization, none of its variants are applicable for mixed mode optimization where some of the variables are continuous and others are integer type. Since the optimization problems arise in some studies involving continuous as well as integer variables, here ABC is modified to deal with mixed mode optimization problems. Basic ABC algorithm starts with a set of randomly generated potential solutions in the search space of the optimization problem under consideration. A solution is treated as a food source and the optimal solution is the best food source. It is an iterative search approach, where each iteration consists of three phases – employee bee phase, onlooker bee phase and scout bee phase. Each food source (potential solution) is associated with an employee bee, i.e., number of solutions equal to the number of employee bees. Each employee bee tries to find a better food source, i.e., tries to find a better neighbour solution using some search strategies. After operations of all the employee bees, i.e., after employee bee phase, onlooker bee phase starts. Number of onlooker bees is fixed and each bee chooses a solution (food source) from the solution set according to their fitness and tries to find a better neighbour solution using some search strategies. After operation of each onlooker bee, scout bee phase starts. Each solution is associated with a counter which keeps record of the consecutive number of iterations in which the solution does not move (improved). If value of this counter exceeds a predefined fixed limit, then it is regenerated in the search space. The algorithm keeps record of the best found solution so far and after the end of the maximum limit of the iterations, the best solution is produced as output. In basic ABC algorithm, only one search strategy was used by the employee bees and the onlooker bees. Kiran et al. [96] proposed a variant of ABC named ABCVSS where multiple solution

update rules are proposed for continuous optimization. Similar to ABCVSS, in this study a set of search strategies (update rules) is suggested and a bee chooses a strategy (update rule) from this set according to the performance of the strategy. Rules are made in such a manner that these are capable of searching neighbour solutions of a solution by perturbing continuous as well as integer variables. The ABC algorithm with the proposed rules is named as Mixed-mode Multi-choice Artificial Bee Colony (MMCABC) Algorithm and is briefly discussed below.

Let $S = \{X_1, X_2, ..., X_n\}$ be the initial swarm consists of n solutions randomly generated in the search space of the optimization problem under consideration and the problem consists of m variables. Among these m variables, m_1 are continuous variables and m_2 are integer variables, i.e., $m = m_1 + m_2$. Without loss of generality let first m_1 variables are continuous and remaining are integer variables. Then the i-th solution X_i consists of m components and is represented by $X_i = (x_{i1}, x_{i2}, ..., x_{im_1}, x_{i(m_1+1)}, x_{i(m_1+2)}, ..., x_{im})$. Best solution found so far is denoted by $X_B = (x_{B1}, x_{B2}, ..., x_{Bm_1}, x_{B(m_1+1)}, x_{B(m_1+2)}, ..., x_{Bm})$. For the movement of a solution, X_i, perturbation is made on a randomly selected component of the solution. Let j-th component x_{ij} of X_i is selected randomly for perturbation. Then X_i perturbed to $X_i^{'} = (x_{i1}, x_{i2}, ..., x_{ij}^{'}, ..., x_{im})$ and perturbation is made by selecting a rule from the following two sets of rules according to the performance of the rules. First set of rules is applicable for the continuous variables and the second set of rules is applicable for the integer variables.

Rule Set-1: A rule in this set is randomly selected for the perturbation of a continuous variable x_{ij} of a potential solution X_i. In the rules, r_1, r_2 are uniformly distributed over $(-1, 1)$ and k, l are randomly selected from the set $\{1, 2, ..., n\}$.

$$x_{ij}^{'} = x_{ij} + r_1(x_{Bj} - x_{ij}) \tag{2.46}$$

$$x_{ij}^{'} = x_{ij} + r_1(x_{kj} - x_{ij}) \tag{2.47}$$

$$x_{ij}^{'} = x_{ij} + r_1(x_{Bj} - x_{ij}) + r_2(x_{kj} - x_{ij}) \tag{2.48}$$

$$x_{ij}^{'} = x_{ij} + r_1(x_{Bj} - x_{ij}) + r_2(x_{kj} - x_{lj}) \tag{2.49}$$

Rule Set-2: A rule in this set is randomly selected for the perturbation of an integer variable x_{ij} of a potential solution X_i. In the rules, r_1, r_2 are two randomly generated integers in the interval specified in the respective rule and k, l are

randomly selected from the set $\{1, 2, ..., n\}$.

$$x'_{ij} = x_{ij} + r_1, \text{ where } r_1 \in \begin{cases} [0, x_{Bj} - x_{ij}], & \text{if } x_{Bj} - x_{ij} \geq 0 \\ [x_{Bj} - x_{ij}, 0], & \text{if } x_{Bj} - x_{ij} \leq 0 \end{cases} \tag{2.50}$$

$$x'_{ij} = x_{ij} + r_2, \text{ where } r_2 \in \begin{cases} [0, x_{kj} - x_{ij}], & \text{if } x_{kj} - x_{ij} \geq 0 \\ [x_{kj} - x_{ij}, 0], & \text{if } x_{kj} - x_{ij} \leq 0 \end{cases} \tag{2.51}$$

$$x'_{ij} = x_{ij} + r_1 + r_2 \tag{2.52}$$

$$x'_{ij} = x_{ij} + r_1 + r_3, \text{ where } r_3 \in \begin{cases} [0, x_{kj} - x_{lj}], & \text{if } x_{kj} - x_{lj} \geq 0 \\ [x_{kj} - x_{lj}, 0], & \text{if } x_{kj} - x_{lj} \leq 0 \end{cases} \tag{2.53}$$

Each rule is associated with a counter which is initialized with 1. If p-th rule is selected for the perturbation of X_i and the perturbed solution X'_i is better than X_i, then X_i is replaced by X'_i and the counter variable corresponding to p-th rule is increased by 1. Selection of a rule is made by Roulette Wheel selection process [93, 123] depending upon the value of its counter variable. This change is incorporated in the basic ABC to create MMCABC and is used to find marketing decisions for some models in this thesis.

Implementation and Testing: With the above function and values the algorithm is implemented using Dev C++ (version 5.11) programming language in a computer with core i3 CPU, 2 GB RAM, 2.53 GHz processor speed and Windows-7 operating system.

To test the performance of the proposed MMCABC algorithm, it is first tested against a set of benchmark test problems having continuous variables only that are available in the literature and are listed in Table 2.10. These test functions are also modified by adding some functions of integer variables and the list of modified functions (MF) are given in Table 2.12. In this table (Table 2.12), the variables x_i $(i = 1, 2, ..., n_1)$ are continuous variables and the variables y_i $(i = 1, 2, ..., n_2)$ are integer variables; where n_1 and n_2 are the number of variables of continuous variables and integer variables respectively. For each of the modified function, the number of variables are considered as $n_1 = 30$ and $n_2 = 10$.

The parametric values of the algorithm used for the purpose are taken as: number of employee bees = 40, number of onlooker bees = 40 and upper limit of the counter of a solution is 1600, i.e., if a solution is not moved by any bee (employee or onlooker) in 1600 attempts, then a scout bee will regenerate it and its counter is set to 0. The maximum number of iterations used to find optimal solution is taken

TABLE 2.10: Benchmark continuous test functions used for the testing of MMCABC

No. of Function	Name	Search Range	Function
F1	Sphere	[-100,100]	$f_1(\vec{X}) = \sum_{i=1}^{n} x_i^2$
F2	Elliptic	[-100,100]	$f_2(\vec{X}) = \sum_{i=1}^{n} (10^6)^{(i-1)/(n-1)} x_i^2$
F3	SumSquares	[-10,10]	$f_3(\vec{X}) = \sum_{i=1}^{n} i x_i^2$
F4	Schwefel 2.22	[-10,10]	$f_4(\vec{X}) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$
F5	Quartic	[-1.28,1.28]	$f_5(\vec{X}) = \sum_{i=1}^{n} i x_i^4$

TABLE 2.11: Results of continuous test functions (for $n = 30$) using MMCABC

Different continuous test Functions	Optimal Objective Value	Mean & S.D. of the objectives obtained by ABCVSS[96]		Mean & S.D. of the objectives obtained by MMCABC using different upper limits of iterations							
				Max iteration = 3000		Max iteration = 4000		Max iteration = 5000		Max iteration = 6000	
		Mean	S.D.	Mean	S.D.	Mean	S.D.	Mean	S.D.	Mean	S.D.
F1	0	1.53E-81	8.37E-81	2.42E-49	4.89E-49	1.78E-67	2.64E-67	1.23E-85	2.07E-85	0	0
F2	0	4.82E-82	2.63E-81	2.12E-48	3.40E-48	1.95E-66	3.62E-66	3.27E-84	7.05E-84	0	0
F3	0	3.19E-89	1.48E-88	1.19E-49	3.38E-49	2.91E-67	8.67E-67	5.66E-86	1.68E-85	0	0
F4	0	7.89E-43	4.32E-42	1.28E-24	2.93E-24	2.35E-33	6.80E-33	7.13E-42	2.12E-41	0	0
F5	0	3.25E-154	1.78E-153	1.89E-100	5.60E-100	5.92E-136	1.40E-135	4.38E-170	0	0	0

as 8000. For each of the test problem of Table 2.10, the algorithm is run 10 times with different seeds of random number generator using different iteration limits and corresponding mean and standard deviation of the obtained objective values are presented in Table 2.11. From Table 2.11 it is found that the algorithm produces exact optimal solutions of all the considered test problems of Table 2.10 in all the runs of the algorithm if the number of iterations exceeds 6000. Performance of the algorithm is compared with a recently published ABC variant, ABCVSS [96]. In Table 2.11, the results of ABCVSS [96] are taken from the corresponding paper. From Table 2.11, it is clear that the performance of MMCABC is comparable to ABCVSS for continuous optimization problems. Again, according to Kiran et al. [96], performance of ABCVSS is better than the existing other ABC variants. From Table 2.11, it is clear that MMCABC produces better optimal solution for the test problems in most of the runs of the algorithm. So the performance of MMCABC is acceptable for continuous optimization problems. Similar to the continuous test problems, the algorithm is also tested against mixed-integer optimization test problems listed in Table 2.12. For each of the test problems of Table 2.12, the algorithm is run 10 times with different seeds of random number generator using different iteration limits and corresponding mean and standard deviation of the obtained objective values are presented in Table 2.13. From Table 2.13, it is also clear that the proposed algorithm is capable of producing results of any mixed-integer programming problem with any desired degree of accuracy.

TABLE 2.12: Mixed integer test functions used for the testing of MMCABC

No. of Function	Search Range	Function
MF1	$x_i \in [-100, 100]$, $y_i \in [0, 10]$	$f_1(\vec{X}) + f_1(\vec{Y}) = \sum_{i=1}^{n_1} x_i^2 + \sum_{i=1}^{n_2} (y_i - i)^2$
MF2	$x_i \in [-100, 100]$, $y_i \in [0, 10]$	$f_2(\vec{X}) + f_2(\vec{Y}) = \sum_{i=1}^{n_1} (10^6)^{(i-1)/(n_1-1)} x_i^2 + \sum_{i=1}^{n_2} (y_i - i)^2$
MF3	$x_i \in [-10, 10]$, $y_i \in [0, 10]$	$f_3(\vec{X}) + f_3(\vec{Y}) = \sum_{i=1}^{n_1} i x_i^2 + \sum_{i=1}^{n_2} (y_i - i)^2$
MF4	$x_i \in [-10, 10]$, $y_i \in [0, 10]$	$f_4(\vec{X}) + f_4(\vec{Y}) = \sum_{i=1}^{n_1} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert + \sum_{i=1}^{n_2} (y_i - i)^2$
MF5	$x_i \in [-1.28, 1.28]$, $y_i \in [0, 10]$	$f_5(\vec{X}) + f_5(\vec{Y}) = \sum_{i=1}^{n_1} i x_i^4 + \sum_{i=1}^{n_2} (y_i - i)^2$

TABLE 2.13: Results of mixed integer test functions (for n_1 = 30 and n_2 = 10) using MMCABC

Test Function	Optimal objective value	Max iteration = 6000			Max iteration = 7000			Max iteration = 8000		
		Mean of objectives	S.D. of objectives	Best objective	Mean of objectives	S.D. of objectives	Best objective	Mean of objectives	S.D. of objectives	Best objective
MF1	0	4.31E-69	9.05E-69	5.19E-80	1.31E-80	2.61E-80	0	0	0	0
MF2	0	9.23E-69	2.77E-68	1.99E-79	2.60E-81	7.79E-81	0	0	0	0
MF3	0	1.14E-71	3.43E-71	4.35E-80	3.00E-84	9.01E-84	0	0	0	0
MF4	0	5.18E-38	1.13E-37	1.14E-40	5.89E-45	1.39E-44	0	0	0	0
MF5	0	3.82E-130	1.15E-129	1.05E-160	4.09E-149	1.23E-148	0	2.63E-171	0	0

MMCABC for Imprecise Environment: In imprecise environment, the solution procedure is same as the MCABC in previous article (cf. § 2.2.2.3).

Constraint Handling: Some models of the thesis consists of constraint optimization. In the presence of constraint, at the time of generation of a neighbour solution/new solution, constraint checking is done by a separate sub-program. If the generated/perturbed solution satisfies the constraints of the problem, then it is included in the solution set, otherwise it is rejected/regenerated. Some problems on mixed integer programming with constraint [40] are presented in Table 2.14. The problems are solved using the proposed approach and the results are presented in Table 2.14. From Table 2.14 it is clear that the algorithm is capable of solving mixed-integer constrained optimization problems.

2.3 Solution Methods/Techniques in Fuzzy Environment

In the most of the programming model, the DM is not able to define the different parameters precisely of the optimization problem under consideration. In these cases, the parameters are either defined in non-stochastic sense, i.e., as fuzzy numbers with feasible membership functions, or in stochastic sense, i.e., as random numbers with feasible probability distributions. In case of non-stochastic sense,

TABLE 2.14: Some problems for constraint handling

No. of Problem	Problem	Exact Solution	Obtained Solution
PR1	Min $f(x,y) = 2x + y$ subject to: $1.25 - x^2 - y \le 0$, $x + y \le 1.6$, $0 \le x \le 1.6$, $y \in \{0,1\}$	$(x,y;f) = (0.5,1;2)$	$(x,y;f) = (0.5,1;2)$
PR2	Min $f(x,y) = -y + 2x - ln(x/2)$ subject to: $-x - ln(x/2) + y \le 0$, $0.5 \le x \le 1.5$, $y \in \{0,1\}$	$(x,y;f) = (1.375,1;2.124)$	$(x,y;f) = (1.375,1;2.124)$
PR3	Min $f(x) = x_1^2 + x_1x_2 + 2x_2^2 - 6x_1 - 2x_2 - 12x_3$ subject to: $2x_1^2 + x_2^2 \le 15$, $-x_1 + 2x_2 + x_3 \le 3$, $0 \le x_1, x_2, x_3 \le 10$, integer variables	$(x_1,x_2,x_3;f)$ $= (2,0,5;-68)$	$(x_1,x_2,x_3;f)$ $= (2,0,5;-68)$
PR4	Min $f(x,y) = (y_1-1)^2 + (y_2-1)^2 + (y_3-1)^2$ $-ln(y_4+1) + (x_1-1)^2 + (x_2-2)^2 + (x_3-3)^2$ subject to: $y_1 + y_2 + y_3 + x_1 + x_2 + x_3 \le 5.0$, $y_3^2 + x_1^2 + x_2^2 + x_3^2 \le 5.5$, $y_1 + x_1 \le 1.2$, $y_2 + x_2 \le 1.8$, $y_3 + x_3 \le 2.5$, $y_4 + x_1 \le 1.2$, $y_2^2 + x_2^2 \le 1.64$, $y_3^2 + x_3^2 \le 4.25$, $y_2^2 + x_3^2 \le 4.64$, $x_1, x_2, x_3 \ge 0$, $y_1, y_2, y_3, y_4 \in \{0,1\}$	$(x_1,x_2,x_3,y_1,y_2,y_3,y_4;f)$ $= (0.2, 1.280624, 1.954483,$ $1,0,0,1;3.557463)$	$(x_1,x_2,x_3,y_1,y_2,y_3,y_4;f)$ $= (0.2, 1.276953, 1.956861,$ $1,0,0,1;3.557787)$

the problem belongs to the class of SOOP in fuzzy environment. A crisp SOOP may be defined as follows:

$$\left.\begin{array}{lll} \text{Max} & f(x,a,b) & \\ \text{subject to} & g_r(x,a) \le c_r, & r = 1,2,...,m \\ & x_k \ge 0, & k = 1,2,...,n \end{array}\right\} \tag{2.54}$$

where, $x = (x_1, x_2, ..., x_n)^T$ is crisp decision vector, $a = (a_1, a_2, ..., a_p)^T$ and $b = (b_1, b_2, ..., b_q)^T$ are crisp parameter vectors, $c = (c_1, c_2, ..., c_m)^T$ is crisp resource vector. There are two types of optimization problems in fuzzy environment.

2.3.1 Type-I: Imprecise (fuzzy) Parameters in Objective Function

If the parameter vector b of the SOOP (Eqn. (2.54)) is fuzzy in nature (i.e., $\tilde{b}$), then the SOOP reduces to a fuzzy single objective optimization problem (FSOOP) as

$$\left.\begin{array}{lll} \text{Max} & \tilde{f}(x,a,\tilde{b}) & \\ \text{subject to} & g_r(x,a) \le c_r, & r = 1,2,...,m \\ & x_k \ge 0, & k = 1,2,...,n \end{array}\right\} \tag{2.55}$$

where, $x = (x_1, x_2, ..., x_n)^T$ is crisp decision vector, $a = (a_1, a_2, ..., a_p)^T$ is crisp parameter vector, $\tilde{b} = (\tilde{b}_1, \tilde{b}_2, ..., \tilde{b}_q)^T$ is fuzzy parameter vector, $c = (c_1, c_2, ..., c_m)^T$ is crisp resource vector.

Solution Procedure 1: Easiest way to deal with the problem (Eqn. (2.55)) is to replace the fuzzy parameters with equivalent crisp parameter, e.g., expected values of fuzzy parameters, GMIV of fuzzy parameters etc. Then the problem reduces to a SOOP and the reduced SOOP can be solved by any suitable solution method. In this thesis, expected values of fuzzy objectives are used in Model 4.1 and GMIV of the fuzzy objectives is used in Model 4.2.

Solution Procedure 2: Another way to deal with the problem (Eqn. (2.55)) is the use of any soft computing technique like PSO, MCABC, MMCABC etc. These algorithms are efficient to find the solution of any crisp optimization problems. But in case of fuzzy optimization problems, there is require fuzzy comparison, i.e., the comparison between two fuzzy objectives. To overcome this situation, the credibility measure (cf. §2.1.2.5) of fuzzy event can be used, which are discussed earlier. In this way one can optimize a fuzzy optimization problem without transferring the problem into its crisp equivalent. In this thesis, some fuzzy models (Models 3.1, 3.2, 4.1, 4.2, 4.3) are optimized using this approach.

2.3.2 Type-II: Imprecise (fuzzy) Parameters in both the Objective Function and Constraint

In the above FSOOP (Eqn. (2.55)), if the parameter vector a and the resource vector c are also fuzzy in nature (i.e., $\tilde{a}$ and $\tilde{c}$), then FSOOP reduces to a fuzzy constraint single objective optimization problem (FCSOOP) as

$$\left.\begin{array}{lll} \text{Max} & \tilde{f}(x, \tilde{a}, \tilde{b}) & \\ \text{subject to} & \tilde{g}_r(x, \tilde{a}) \leq \tilde{c}_r, & r = 1, 2, ..., m \\ & x_k \geq 0, & k = 1, 2, ..., n \end{array}\right\} \tag{2.56}$$

where, $x = (x_1, x_2, ..., x_n)^T$ is crisp decision vector, $\tilde{a} = (\tilde{a}_1, \tilde{a}_2, ..., \tilde{a}_p)^T$ and $\tilde{b} = (\tilde{b}_1, \tilde{b}_2, ..., \tilde{b}_q)^T$ are fuzzy parameter vectors, $\tilde{c} = (\tilde{c}_1, \tilde{c}_2, ..., \tilde{c}_m)^T$ is fuzzy resource vector.

In real life problem, it is often observed that the α-cut $[f_L(x,a,b,\alpha), f_R(x,a,b,\alpha)]$ of the fuzzy objective $\tilde{f}(x,\tilde{a},\tilde{b})$ is available rather its explicit form:

$$\left.\begin{array}{lll} \text{Max} & [f_L(x,a,b,\alpha), f_R(x,a,b,\alpha)] & \\ \text{subject to} & \tilde{g}_r(x,\tilde{a}) \leq \tilde{c}_r, & r = 1,2,...,m \\ & x_k \geq 0, & k = 1,2,...,n \end{array}\right\} \tag{2.57}$$

PSO, MCABC, MMCABC can be used to solve the problems (Eqn. (2.56)), where the objective function can be dealt with the similar way of solution procedure 2 of type-1. To deal with the fuzzy constraint $\tilde{g}_r(x,\tilde{a}) \leq \tilde{c}_r$, the credibility measure approach can be used. Using this approach, the value x_0 of the decision vector x is feasible if and only if $Cr(\tilde{g}_r(x_0,\tilde{a}) \leq \tilde{c}_r) > 0.5$ (cf. Lemma 2.4).

The same techniques can be used to solve the problems (Eqn. (2.57)), where the constraints can be dealt with the same approach as stated above. As no well established comparison approach for α-cut of fuzzy objectives is available in the literature, fuzzy preference ordering (cf. § 2.1.5) of interval comparison is used to find the marketing decisions. In this thesis, this approach is followed to solve the fuzzy inventory model (Model 3.3).

2.4 Solution Methods/Techniques in Rough Environment

2.4.1 Type-I: Imprecise (rough) Parameters in Objective Function

If the parameter vector b of the SOOP (Eqn. (2.54)) is rough in nature (i.e., $\check{b}$), then the SOOP reduces to a rough single objective optimization problem (RSOOP) as

$$\left.\begin{array}{lll} \text{Max} & \check{f}(x,a,\check{b}) & \\ \text{subject to} & g_r(x,a) \leq c_r, & r = 1,2,...,m \\ & x_k \geq 0, & k = 1,2,...,n \end{array}\right\} \tag{2.58}$$

where, $x = (x_1, x_2, ..., x_n)^T$ is crisp decision vector, $a = (a_1, a_2, ..., a_p)^T$ is crisp parameter vector, $\check{b} = (\check{b}_1, \check{b}_2, ..., \check{b}_q)^T$ is rough parameter vector, $c = (c_1, c_2, ..., c_m)^T$ is crisp resource vector.

Solution Procedure 1: Easiest way to deal with the problem (Eqn. (2.58)) is to replace the rough parameters with equivalent crisp parameter, e.g., expected values of rough parameters. Then the problem reduces to a SOOP and the reduced SOOP can be solved by any suitable solution method. In this thesis, expected values of rough objectives are used in Model 4.1.

Solution Procedure 2: Another way to deal with the problem (Eqn. (2.58)) is the use of any soft computing technique like PSO, MCABC, MMCABC etc. These algorithms are efficient to find the solution of any crisp optimization problems. But in case of rough optimization problems, there is require rough comparison, i.e., the comparison between two rough objectives. To overcome this situation, the trust measure (cf. § 2.1.3) of rough event can be used, which are discussed earlier. In this way one can optimize a rough optimization problem without transferring the problem into its crisp equivalent. In this thesis, some rough models (Models 3.1, 4.3) are optimized using this approach.

2.4.2 Type-II: Imprecise (rough) Parameters in both the Objective Function and Constraint

In the above RSOOP (Eqn. (2.58)), if the parameter vector a and the resource vector c are also rough in nature (i.e., $\check{a}$ and $\check{c}$), then RSOOP reduces to a rough constraint single objective optimization problem (RCSOOP) as

$$\left.\begin{array}{lll} \text{Max} & \check{f}(x, \check{a}, \check{b}) & \\ \text{subject to} & \check{g}_r(x, \check{a}) \leq \check{c}_r, & r = 1, 2, ..., m \\ & x_k \geq 0, & k = 1, 2, ..., n \end{array}\right\} \tag{2.59}$$

where, $x = (x_1, x_2, ..., x_n)^T$ is crisp decision vector, $\check{a} = (\check{a}_1, \check{a}_2, ..., \check{a}_p)^T$ and $\check{b} = (\check{b}_1, \check{b}_2, ..., \check{b}_q)^T$ are rough parameter vectors, $\check{c} = (\check{c}_1, \check{c}_2, ..., \check{c}_m)^T$ is rough resource vector.

PSO, MCABC, MMCABC can be used to solve the problems (Eqn. (2.59)), where the objective function can be dealt with the similar way of solution procedure 2 of type-1. To deal with the rough constraint $\check{g}_r(x, \check{a}) \leq \check{c}_r$, the trust measure

approach can be used. Using this approach, the value x_0 of the decision vector x is feasible if and only if $Tr(\check{g}_r(x_0, \check{a}) \leq \check{c}_r) > 0.5$ (cf. Lemma 2.8).